ESCUELA SUPERIOR DE INGENIERÍA

2º CICLO INGENIERÍA EN INFORMÁTICA

IMPLANTACIÓN Y MEJORA DEL SISTEMA DE COLABORACIÓN MULTIMEDIA ACCESS GRID

Jesús Cea Oliva

16 de septiembre de 2010

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO EN INFORMÁTICA

IMPLANTACIÓN Y MEJORA DEL SISTEMA DE COLABORACIÓN MULTIMEDIA ACCESS GRID

- Departamento: Lenguajes y sistemas informáticos
- Directores del proyecto: Daniel Molina Cabrera - Manuel Palomo Duarte
- Autor del proyecto: Jesús Cea Oliva

Cádiz, 16 de septiembre de 2010

Fdo: Jesús Cea Oliva

Agradecimientos

Me gustaría agradecer y/o dedicar este proyecto a varias personas.

En primer lugar quiero agradecer a Miguel Ángel Pérez Prado, que vivió y sufrió conmigo durante toda esta aventura, aprendimos mucho el uno del otro.

También me gustaría agradecer este proyecto a Manuel Palomo y a Daniel Molina que, además de hacer de tutores de proyecto, a veces han hecho el papel de psicólogo en los malos momentos.

Por otro lado me gustaría también agradecer la gran ayuda prestada varias personas de la Universidad de Cádiz, Juan Luis Jiménez que, aunque en más de una ocasión nos hemos desesperado, siempre ha estado ahí encima de nosotros para que hiciéramos un buen trabajo. A Ignacio Pérez, que nos ayudó mucho sobre todo en nuestros comienzos y siempre ha estado ahí cada vez que necesitábamos su ayuda. A José Luis, que cada vez que hemos tenido problemas con las redes ha estado soportando todos nuestros problemas. Y por supuesto, a todo el personal de los Campus de Cádiz (Miguel Zea y Miguel Bolaño), del Campus de Algeciras (Jesús Franco y Alberto Palomares) y del Campus de Jerez (Julián Figueroa) quienes han estado soportando todos nuestros quebraderos de cabeza y siempre han estado ahí para realizar todo tipo de pruebas y aportar su ayuda siempre que lo necesitábamos.

Por último, me gustaría agradecer, por parte de la Universidad de Cádiz, a Ambrosio Díaz y a Eduardo Blanco, y por parte de Auditel Ingenería y Servicios, a Waldo Franco, a Javier Sobrino, a Antonio Benítez y, en general, a todo el personal de Auditel, por ofrecernos esta gran experiencia, nueva para mí, y por la ayuda prestada.

Por último, me gustaría dedicar este proyecto a mi familia y a mis amigos, que también han hecho de psicólogos y han aguantado mis malos y mis buenos momentos ante este proyecto, recibiendo siempre, por parte de ellos un gran apoyo.

Licencia

Notación y formato

Aquí incluiremos los aspectos relevantes a la notación y el formato a lo largo del documento. Para simplificar podemos generar comandos nuevos que nos ayuden a ello, ver `comandos.sty` para más información.

Cuando nos refiramos a un programa en concreto, utilizaremos la notación:
Access Grid.

Cuando nos refiramos a un comando usaremos la notación:
`dir`, `cp`.

Cuando nos refiramos a una clase, o función de un lenguaje, usaremos la notación:
`CmdlineApplication`.

Cuando nos refiramos a una serie de comandos escrito en una terminal usaremos la notación:

```
openssl x509 -CA ca_cert.pem -CAkey ca_key.pem -req
    -in petic-cert-client.pem -set_serial 3 -days 15
    -extfile config_client.signing_policy -sha1
    -out client-cert.pem
```

Índice general

Indice de figuras

Capítulo 1

Motivación y contexto del proyecto

1.1. Introducción

El mundo de las telecomunicaciones avanza a pasos agigantados y ha supuesto ser toda una revolución, ya que, con pocos recursos, podemos comunicarnos con cualquier persona, esté donde esté, y, con cada vez más, formas diferentes.

El trabajo en grupo sufre cambios continuos para adaptarse a las circunstancias actuales y se apoya, en la mayoría de los casos, en nuevas formas de comunicación con las cuales se transmite la información. Está demostrado que el conocimiento se da en mayor medida y velocidad cuando se unen esfuerzos y se tienen bien definidos los objetivos.

La especialización en cada una de las áreas del conocimiento ha diversificado, en cantidad y en geografía, los diversos grupos de personas que llevan a cabo desarrollos afines. Cada vez es más común que, cada cierto tiempo, se realicen eventos tales como congresos, foros de debate, etc donde, en la mayoría de los casos, implican que las personas que deseen asistir o participar a dichos eventos tengan que desplazarse, además de otros factores tales como la logística de organización y gastos que dependerán del desplazamiento físico de personas y/o equipo a grado tal, que grandes eventos impliquen enormes esfuerzos en todos los sentidos.

En los últimos diez años, con el crecimiento de Internet, las distancias y tiempos de comunicación se han acortado; de manera adicional, se han sumado herramientas como el correo electrónico, la publicación electrónica de información en páginas de Internet y la creación de grupos de discusión con temas específicos, etc. La comunidad científica y, en general, la comunidad docente, han incorporado de forma gradual estos elementos en el desarrollo de sus actividades cotidianas. Uno de estos elementos es la videoconferencia, la cual es sumamente utilizada en nuestros días para muchos fines.

La videoconferencia es la comunicación simultánea bidireccional de audio y vídeo, permitiendo mantener reuniones con grupos de personas situadas geográficamente en lugares diferentes. Adicionalmente, pueden ofrecerse facilidades telemáticas o de otro tipo como el intercambio de informaciones gráficas, imágenes fijas, transmisión de ficheros desde el pc, etc. Su implementación proporciona importantes beneficios: como ya se puede suponer de lo anteriormente dicho, uno de los beneficios es la posibilidad de comunicarse entre varias personas que están en lugares geográficamente distintos. Consecuentemente, otro de los beneficios es la posibilidad de llevar a cabo un trabajo colaborativo entre dichas personas, sin necesidad de que tengan que verse físicamente en un sitio[1].

[1] Imagen cogida de `http://www.lbt.es/Videoconferencia`

Figura 1.1: Ejemplo de una videoconferencia

La Universidad de Cádiz apuesta por estas tecnologías y es una de las universidades pioneras en el ámbito de la teledocencia. Gracias a la empresa Auditel, han creado tres salas de teledocencia en los Campus de Puerto Real, Cádiz y Algeciras. Dichas salas se utilizarán para fines relacionados con la docencia: impartir clases a distancia, colaborar en un grupo de investigación donde cada miembro del grupo está en un lugar diferente, reuniones a distancia, congresos, etc.

Figura 1.2: Sala de Teledocencia del Campus de Puerto Real

Dichas salas han sido equipadas con la última tecnología tanto en hardware como, por ejemplo,

audio, vídeo, domótica, entre otros ámbitos; como en software, obteniendo un sin fin de posibilidades a la hora de utilizar dicho equipamiento. De los distintos usos posibles para este tipo de salas, podemos identificar algunos de ellos:

- Utilización del equipamiento como sala de debate con distribución de sonido en sala.
- Sesiones de docencia con audio y vídeo.
- Transmisión de vídeo y audio a través de un codificador de MPEG-2 y/o un de Codificador Windows Media a Internet para la visualización de usuarios remotos.
- Realización de Workshops y otros eventos similares.

Para la videoconferencia, y, en general, como sistema de colaboración, se ha optado por utilizar el sistema *Access Grid*.

Sin embargo, tanto la Universidad de Cádiz, como la empresa Auditel, al ser pioneros en estas tecnologías, surgieron varios problemas a la hora de poner en funcionamiento las Salas.

En primer lugar, en la Universidad de Cádiz no existía personal cualificado para configurar y utilizar correctamente el equipamiento instalado en las salas. La empresa Auditel impartió un curso de formación para algunos empleados, pero no fue suficiente, debido a la cantidad de equipamiento existente y la gran cantidad de posibilidades que ofrecen tanto dicho equipamiento, como el sistema *Access Grid*.

Además, la documentación del sistema *Access Grid* es escasa y, a nivel nacional, prácticamente inexistente, ni si quiera existen foros de discusión donde preguntar dudas relacionadas con este sistema, por lo que, debido a ésto, el personal de la Universidad de Cádiz no sabía utilizar dicho sistema.

Por último, la aparición de nuevas necesidades futuras, tanto a nivel de implantación (instalar nuevos equipamientos, etc.), como a nivel de desarrollo (desarrollar aplicaciones de apoyo y mejora) fueron determinantes para percatarse de que hacía falta personal dedicado a estas salas.

1.2. Objetivo

Visto los problemas surgidos en el apartado anterior, mi compañero Miguel Ángel Pérez Prado y yo, fuimos convocados para formar parte, como becarios de prácticas de empresa, de la empresa Auditel, con el objetivo de formarnos y solventar todos los problemas comentados.

Así, pues, el objetivo de este Proyecto de Final de Carrera es documentar, configurar y desarrollar el sistema *Access Grid* para su implantación en aulas de teledocencia, así como en distintas salas de distintos estamentos para reuniones de grupos.

Se documentó tanto todo el proceso de implantación, como el propio sistema *Access Grid*, detallando cada una de las funciones que ofrece, cómo configurar el sistema correctamente, valores óptimos de configuración, etc. En definitiva, se hizo un manual completo de *Access Grid*.

Se configuró y optimizó el sistema *Access Grid* en todas las Salas de Teledocencia instaladas en el Campus de Cádiz, para obtener el mejor funcionamiento y el mejor aprovechamiento de las Salas posibles.

En cuanto al desarrollo, el sistema *Access Grid* es software libre y de código abierto, es decir, podemos descargarnos su código fuente para visualizarlo o editarlo.

Por otro lado, *Access Grid* dispone de una API para desarrollar **aplicaciones compartidas** (en inglés, *Shared Applications*). Las aplicaciones compartidas son programas que se ejecutan bajo el entorno *Access Grid*, de forma que dichas aplicaciones se ejecutan a todos los participantes que estén conectados en la Sala y cualquier cambio o efecto que se produzca durante la ejecución del software, afectará al resto de participantes conectados.

Se han detectado algunos fallos del sistema *Access Grid* durante su ejecución. Estos errores se solucionaron y se notificó al equipo de desarrollo de *Access Grid*. Por último, se desarrollaron aplicaciones compartidas para mejorar la experiencia del sistema *Access Grid*, así como para optimizar el uso de las Salas de Teledocencia.

1.3. Estructura del documento

La documentación de este Proyecto de Final de Carrera se divide en varios apartados que, a continuación, es pasará a explicar.

1. **Estado del Arte:** En este apartado se hablará sobre qué es Access Grid, junto a sus características y funcionalidades. Además se hará una comparativa del sistema Access Grid con otros sistemas similares. Por último se explicará por qué se ha decantado por Access Grid.
2. **Desarrollo del Proyecto:** En este apartado se explicará la planificación realizada para poder llevar a cabo la realización de este Proyecto de Final de Carrera.
3. **Fase de Implantación:** En este apartado se detallará cómo, partiendo desde cero, se ha conseguido implantar el sistema Access Grid en las Salas de Teledocencia del Campus de Cádiz, así como su configuración, optimización y documentación.
4. **Fase de Desarrollo:** En este apartado se explicará todo el proceso de desarrollo tanto la corrección y mejora del código del sistema de Access Grid, como el desarrollo de aplicaciones compartidas. Además se explicarán las aplicaciones desarrolladas, incluyendo la documentación correspondiente a las fases de análisis y diseño de cada una de las aplicaciones desarrolladas.
5. **Conclusiones:** En este apartado se detallarán las conclusiones obtenidas a lo largo del desarrollo de este Proyecto de Final de Carrera. Además se hablará sobre los futuros proyectos que se pretenden hacer relacionados con este proyecto.
6. **Apéndices:** La documentación de este Proyecto de Final de Carrera incluye los siguientes apéndices:
   - **Eventos realizados:** Como muestra de que se han superado todos los objetivos establecidos para este Proyecto de Final de Carrera, se hablará de los eventos que se han llevado a cabo durante estos meses.
   - **Software y Librerías Usadas:** En este apartado se explicará qué software y librerías han sido necesarias utilizar para el desarrollo del proyecto.
   - **Documentación generada:** En este apartado se incluirá toda la documentación que se ha ido desarrollando y generando a lo largo de estos meses.
7. **Bibliografía y referencias:** Conjunto de bibliografía, artículos y enlaces webs consultados para poder realizar este Proyecto de Final de Carrera.

Capítulo 2

Estado del Arte

A continuación, en este capítulo, se hablará sobre el estado del arte de este Proyecto de Final de Carrera. Se explicará qué es *Access Grid*, así como sus características. Además, se hablará de otros sistemas semejantes a *Access Grid*, comparándolos con este sistema y explicando por qué se ha decantado por él.

2.1. ¿Qué es Access Grid?

Access Grid es un entorno integrado que soporta la comunicación de grupos utilizando redes de alta velocidad a través de Internet. *Access Grid* es un conjunto de recursos, incluyendo grandes pantallas multimedia, presentación y entornos interactivos y la compartición de aplicaciones en sus entornos de visualización. Estos recursos se utilizan para apoyar las interacciones entre grupos a través de la red. Ofrece audio de alta calidad y vídeo en tiempo real, que permite a grupos situados en varios lugares, interactuar y compartir datos e instrumentos científicos al mismo tiempo. [1]

Figura 2.1: Logo de Access Grid

Access Grid, además de servir para realizar reuniones entre individuos o grupos utilizando recursos de audio y vídeo, permite compartir aplicaciones simultáneamente entre los participantes, ya sea una presentación de *Powerpoint*, un documento en PDF, nuestro propio escritorio o un navegador web mientras estamos en Internet.

2.2. Una breve historia

Access Grid fue desarrollado por **FuturesLab** en el Laboratorio Nacional de Argonne, Chicago. Su primera aparición y primer evento a gran escala fue en el año 1999, en una serie de conferencias, llamado **"La Alliance Chautauqua 99"** que duró dos días, sobre la ciencia computacional organizada por la **NCSA**. *Access Grid* fue más tarde dado a conocer al público internacional en el evento **"Supercomputing '99 "** en Portland.

[1] Imagen cogida de [Agr10]

El primer nodo *Access Grid* de la costa oeste de Estados Unidos se instaló en el San Diego Supercomputer Center (UC San Diego) en enero de 2000 por B. Pailthorpe, J Moreland y N Bordes.

El 23 y 24 de marzo de 2000, se usó para albergar una reunión del West Coast / Washington DC President's Information Technology Advisory Committee (PITAC), donde participaron treinta CEOs de la costa oeste, quienes no tuvieron que desplazarse a Washington DC para tal reunión.

Posteriormente, *Access Grid* se extendió hacia Europa. *Access Grid* está soportado en la comunidad académica del Reino Unido, por el Centro de Soporte de *Access Grid* (Access Grid Support Centre, AGSC), fundado por JISC y gestionado por JANET.

El primer nodo *Access Grid* europeo fue construido en la Universidad de Manchester en 2001 y, posteriormente su AGSC comenzó en Abril de 2004.

Actualmente existen cerca de 300 nodos de *Access Grid* con AGSC en el Reino Unido, desde Salas completamente equipadas hasta pequeñas salas de reuniones o despachos.

Existe un gran número de proyectos académicos usando la tecnología *Access Grid*, como los proyectos matemáticos Taught Course Centre y MAGIC.

2.3. Características de Access Grid

Access Grid, al tratarse de un sistema colaborativo, incluye una serie de características generales:

- Sistema de Multiconferencia con audio y vídeo, permitiendo establecer contacto, tanto visual, como auditivo, con varios participantes simultáneamente.

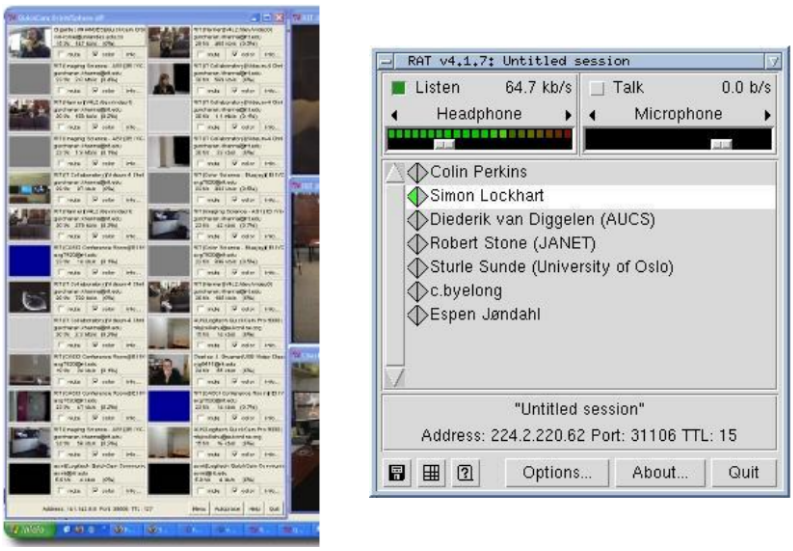


Figura 2.2: Multiconferencia en Access Grid

- Capacidad para codificar en varios formatos y códecs, tanto en audio como en vídeo, tales como H.261, H.264, MPEG-4 e incluso formatos HD.
- Sistema de compartición de archivos entre los participantes.

- Sistema de chat entre los participantes.
- Conexiones multicast y unicast (a través de Puentes Multicasts), debido a la gran cantidad de información que viaja en una sesión de *Access Grid*, el sistema está ideado para trabajar bajo redes multicast, para no sobrecargar la red. En aquellos lugares donde no se disponga de Multicast, *Access Grid* permite utilizar Puentes Multicast, es decir, se conectaría al puente por Unicast, y dicho puente transmitiría toda la información por Multicast.
- Uso de aplicaciones compartidas. *Access Grid* integra varias aplicaciones tales como un navegador web compartido o una pizarra compartida, permitiendo, por un lado, que todos los participantes puedan interactuar en dicha aplicación como si fuera una sola y, por otro, transmitir cualquier cambio al resto de participantes al instante.
- Extensible. *Access Grid* dispone de una API para desarrolladores para corregir o mejorar el Sistema o para crear nuevas aplicaciones compartidas.

2.4. Arquitectura de Access Grid

Para comprender mejor el sistema Access Grid se explicará brevemente su arquitectura. El Sistema consta, básicamente, de dos componentes:

- **El Servidor de Salas:** Encargado de conectar a los clientes entre sí, habilitando un punto de encuentro común. Dentro de un mismo servidor de salas pueden existir tantas salas como sean necesarias. Dichas salas pueden ser públicas o privadas.
- **El Cliente de Sala:** Son los participantes en las reuniones. Un cliente puede ser desde un usuario con un portátil hasta una sala de conferencias completamente equipada con equipos de audio y vídeo de alta definición.

Los elementos principales que forman dicha arquitectura son:

- **VenueServer:** Servidor de Salas
- **Venue:** Una sala
- **VenueClient:** Cliente de Sala
- **Node:** Un Nodo

La siguiente figura muestra cómo se conectan estos elementos formando la estructura de Access Grid:

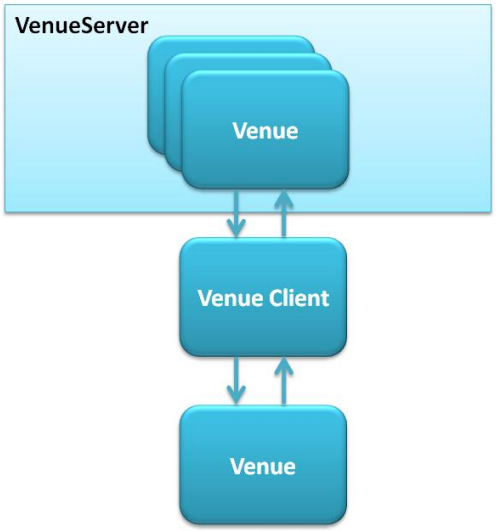


Figura 2.3: Arquitectura de Access Grid

El Servidor de Salas (**VenueServer**) gestiona un número de Salas (**Venues**), al igual que un servidor web gestiona páginas web. El Cliente de Sala (**VenueClient**), por un lado se conecta a una Sala y luego se comunica con el Nodo para realizar una determinada acción en dicha Sala; por ejemplo, iniciar el servicio de audio para la Sala en la que se encuentra.

2.4.1. Arquitectura del Nodo

El elemento Nodo contiene una colección de componentes software ejecutándose en un Cliente de Sala para controlar herramientas que reciben y emiten audio y vídeo, entre otras cosas.

El Cliente de Sala comunica al Servicio de Nodo (**NodeService**) de cambios de estado. El Servicio de Nodo agrega los llamados **ServicesManagers**, que son los que se encargan de gestionar los Servicios en las máquinas locales, y recoge información sobre los **Servicios** que se ejecutan en ellas pudiéndose comunicar con los Servicios cuando sea necesario.

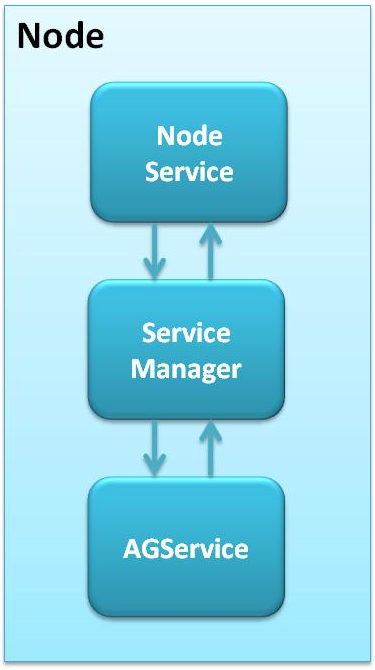


Figura 2.4: Arquitectura de un Nodo Access Grid

Así, pues, vemos que los componentes que forman la arquitectura del Nodo son:

- **NodeService:** Servicio de Nodo. Agrega ServicesManagers de máquinas locales, además de guardar configuraciones en las que se incluyen los servicios, sus descripciones y sus configuraciones individuales.
- **ServiceManager:** Gestor de Servicios. Se encarga de localizar los recursos disponibles en la propia máquina, y en máquinas locales. Además, se encarga de gestionar los servicios.
- **Services:** Los propios Servicios. Lee y escribe configuraciones, como por ejemplo el número y el tipo de la tarjeta de vídeo, el códec de vídeo a usar, la calidad, formato de pantalla, formato del audio, calidad de audio, etc., además, de describir las características de los propios servicios. Por último, ejecuta software subyacente (como *vic* para los servicios de vídeo o *RAT* para los servicios de audio).

Todos los elementos pueden ser llamados a través de la interfaz *SOAP*.

2.4.2. Flujo de Datos al acceder a una Sala

Cuando el Cliente de Sala entra en una Sala se realizan una serie de operaciones para establecer las direcciones de los medios conectados:

- Consulta al Servicio de Nodo las Funcionalidades que ofrecen sus Servicios.
- Transporta dichas Funcionalidades a la Sala (fase de Negociación de Funcionalidades, en inglés **NegotiateCapabilities**).
- La Sala (en la etapa de Negociación de Funcionalidades) asocia las Funcionalidades de entrada con las Funcionalidades de los medios de comunicación conectados que ya han sido establecidos. Si no se encuentran ajustes, la Sala asigna una nueva dirección al medio de comunicación. El

conjunto de direcciones de los medios de comunicación es devuelto al Cliente de Sala (como **StreamDescriptions**).

- El Cliente de Sala pasa estos StreamDescriptions al Servicio de Nodo, que los coloca en servicios individuales basándose en dichas Funcionalidades. Por ejemplo, el stream de audio se pasa al Servicio de Audio, porque tiene una Funcionalidad de tipo *"audio"*.

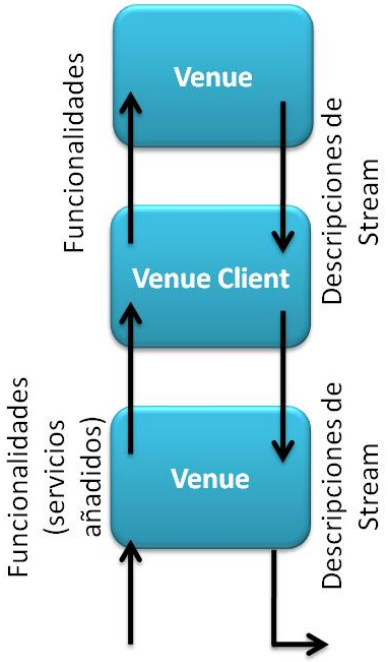


Figura 2.5: Flujo de Datos al acceder a una Sala

2.5. Otros Sistemas de Colaboración

Access Grid no es el único sistema de colaboración multimedia existente, existen varios, cada uno con sus ventajas y sus inconvenientes. A continuación detallaremos los más destacados:

2.5.1. Sistemas de Telepresencia Polycom

El sistema Polycom [2] [3] se podría decir que es uno de los primeros sistemas de multiconferencia.

Polycom es una empresa multinacional con más de 2.700 empleados en el mundo. La compañía fabrica y vende soluciones de comunicaciones por telepresencia y por voz. Se introdujeron en el mundo de la videoconferencia en el año 1998. Polycom introdujo la línea de productos *ViewStation* que tenía como características capacidades multipunto (comunicación con más de dos usuarios), capacidad de compartir contenido y soporte para utilizar el protocolo de red IP H.323.

En el año 2000, Polycom introdujo una solución de videoconferencia para uso personal llamado *ViaVideo*. Estaba compuesto, esencialmente, por una webcam con capacidad de procesamiento integrado, para que los PCs de escritorio y portátiles de por entonces no sufrieran una carga de trabajo. Con el

[2] `http://www.polycom.es/`
[3] `http://en.wikipedia.org/wiki/Polycom`

paso del tiempo, la potencia de procesamiento creció, Polycom pasó de la solución de escritorio a una solución software llamada *Polycom PVX*.

En 2001, Polycom adquirió Accord Networks y PictureTel. Posteriormente, Polycom ha ido desarrollando nuevas tecnologías y, por lo tanto, nuevos sistemas de videoconferencia integrando dichas tecnologías, como por ejemplo, en el año 2006, Polycom introdujo su primer sistema de videoconferencia en alta definición [4].

Figura 2.6: Un sistema Polycom

Un sistema Polycom es muy fácil de instalar y de usar. Basta con conectar el dispositivo a la red, a través de un cable Ethernet, y a un televisor (o a una pantalla cualquiera) a través de un cable de vídeo. Al conectarse a la red, al sistema se le asigna una dirección IP. Así, cuando se desee realizar una reunión a través de *Polycom*, basta con marcar la dirección IP del usuario que deseemos conectar. Si la reunión consta de varias personas, todos los participantes se conectarían a una dirección IP [5].

Figura 2.7: El sistema Polycom en funcionamiento

Un sistema *Polycom* incorpora, además del componente de marcación, una webcam de alta calidad y un sistema de microfonía integrado en el propio marcador, donde todos los participantes pueden hablar, a distancia, y el sistema de microfonía capturar y emitir el sonido a una calidad excelente.

[4]Imagen cogida de `http://www.fonicapbx.com/`
[5]Imagen cogida de `http://www.digitalmarketla.com/2009/12/3289`

Ventajas e inconvenientes

El sistema *Polycom* posee una serie de ventajas, pero también una serie de inconvenientes. Como ventajas podemos destacar:

- Sistema de multiconferencia de alta calidad tanto en audio como en vídeo.
- La empresa Polycom es, actualmente, una empresa madura, por lo que lleva varios años en el mundo de la videoconferencia y, por tanto, posee una experiencia que otras empresas no dispone.
- El sistema incorpora una webcam y un sistema de microfonía de debate, ahorrando así los costes de adquisición de cámaras de vídeo, sistemas de audio y de uno o varios PCs interconectados.
- Ideal para salas pequeñas de reuniones o despachos.

Sin embargo, los sistemas *Polycom* tienen una serie de desventajas de las cuales, podemos destacar las siguientes:

- El sistema no es libre ni gratuito, es un sistema propietario y cerrado, de pago, además de poseer un **alto coste de adquisición**. Concretamente, un sistema *Polycom* que permite conectarse hasta 5 personas cuesta aproximadamente 350€.
- El sistema está **limitado a un número máximo de participantes**, dependiendo del modelo adquirido. Existen modelos más económicos, pero que únicamente permiten la comunicación entre dos personas, pasando por modelos que permiten la comunicación de hasta cinco personas, y terminando por modelos muchos mas caros, que permiten la comunicación de una decena de personas.
- El sistema es poco estable, a las pocas horas de conexión de una sesión, el sistema suele caerse, desconectando a los participantes de dicha reunión. Los técnicos deben de reiniciar la conexión completamente, con la pérdida de trabajo y de tiempo que ello conlleva.
- Como se ha comentado en las ventajas, el sistema es ideal para salas pequeñas de reuniones o despachos. Los motivos son varios; por un lado, únicamente permite conectar una sola webcam y, por tanto, emitir una sola fuente de vídeo al resto de participantes. En Salas grandes, como las instaladas en la Universidad de Cádiz, disponen de varias pantallas interconectadas a un equipo, estas pantallas funcionan como un escritorio extendido. Así, lo ideal en estas salas, es poder repartir todas las fuentes de vídeo entre todas las pantallas. Sin embargo, los sistemas *Polycom* ésto no lo permiten, teniendo que utilizar una de las pantallas de la Sala. Por otro lado, el audio que ofrecen los sistemas *Polycom* puede ser muy limitado para una Sala de grandes dimensiones y, por tanto, puede oírse muy bajo. Sería necesario agregar al sistema un sistema de audio para amplificar el sonido y adecuarlo a la Sala, con los costes que ésto supone.

2.5.2. Adobe Connect

Adobe® Connect™ [6] [7] es un producto de la empresa **Adobe**. Es una derivación del producto anteriormente conocido como *Macromedia Breeze*, es un sistema de comunicación Web seguro y flexible que permite a los profesionales de TI ampliar y complementar la funcionalidad de *Adobe Acrobat® Connect Professional* para proporcionar soluciones de comunicación Web empresarial para formación, marketing, conferencias Web empresariales y colaboración en línea.

[6] `http://www.adobe.com/es/products/connect/`
[7] `http://en.wikipedia.org/wiki/Adobe_Connect`

El producto está completamente desarrollado con la tecnología *Adobe Flash*, para los clientes, basta únicamente con instalar el respectivo plugin del reproductor de *Flash* para permitir su ejecución y, si dispone de acceso (bien porque el acceso es libre o porque dispone de una cuenta de acceso) inmediatamente entrará en una Sala e iniciará una sesión de *Adobe Connect*.

Estas soluciones están disponibles como suscripción alojada para una gestión sin problemas, o como software con licencia que puede implantarse bajo la protección de servidores de seguridad [8].

Figura 2.8: Adobe Connect en funcionamiento

Todas las Salas de *Adobe Connect* están organizadas por unos elementos llamados 'pods', cada pod desarrolla un rol específico como, por ejemplo, un chat, una pizarra electrónica, un bloc de notas, una presentación, etc.

Para gestionar los usuarios y las salas registradas, junto a otra información (ficheros compartidos, etc.), se utiliza un Sistema de Gestión de Base de Datos, se recomienda utilizar *Microsoft SQL Server*.

El producto se comercia de varias formas distintas: como un producto ya instalado en un equipo, como una aplicación basada en la nube alojada en los servidores de Adobe, o como un servicio gestionado, donde una entidad asociada a Adobe desarrolla y gestiona cada instalación personalizada de forma individual.

Adobe Connect fue desarrollado a partir de otras aplicaciones. Comenzó con la aplicación *Presedia*, creada por **Presedia Publishing System**, cuya característica principal era que incluía una primera versión de un plugin que convertía documentos en formato PowerPoint en formato Flash, que, posteriormente se convertiría en *Adobe Presenter* y en un módulo de entrenamiento. Macromedia adquirió *Presedia* y le añadió un componente para videoconferencia en tiempo real por web, llamado *Breeze Live*, posteriormente renombrado a *Breeze Meeting*.

En la versión 5, *Macromedia Breeze* incluyó cuatro aplicaciones más: *Breeze Presenter*, *Breeze Training*, *Breeze Meeting* y *Breeze Events*. Como posteriormente Macromedia fue adquirida por Adobe, la aplicación *Breeze Meeting* fue renombrada a *Adobe Connect*. La versión completa de este producto incluye versiones renombradas de *Breeze Training*, *Breeze Meeting*, *Breeze Events* y *Breeze Presenter*. Incluso existe una versión "hermana" llamada *Adobe ConnectNow*.

[8] Imagen cogida de `http://www.pdfzone.com/c/a/Authoring/Acrobat-8/4/`

Características de Adobe Connect

Adobe Connect Pro incluye las siguientes características:

- Salas de reuniones ilimitadas y personalizables.
- Múltiples Salas de reuniones por usuario.
- VoIP.
- Integración de Audio.
- Videoconferencia.
- Grabación de las reuniones para, posteriormente, poder visualizarlas.
- Compartición del escritorio al resto de participantes.
- Bloc de notas, chat y pizarra electrónica compartida.
- Gestión de usuarios y administración del sistema para gestionar el acceso de usuarios a las salas y gestionar las propias salas.
- Posibilidad de desarrollar encuestas y tipo tests.

Ventajas e inconvenientes de Adobe Connect

A primera vista, observando las principales características que ofrece *Adobe Connect*, parece que ofrece prácticamente lo mismo, o incluso más funcionalidades que el sistema *Access Grid*. Efectivamente, *Adobe Connect* ofrece una serie de ventajas, de las que podemos destacar las siguientes:

- Sistema de multiconferencia ofreciendo audio y vídeo de alta calidad.
- Numerosas de aplicaciones integradas para complementar al sistema, tales como la pizarra compartida, el visor de escritorios compartidos o el visor de documentos. Además de incluir nuevas aplicaciones no existentes en *Access Grid* tales como el poder generar encuestas y exámenes tipo tests.
- Sistema muy fácil de usar. Para los clientes de una sesión de *Adobe Connect* basta con introducir en su navegador web la dirección del Servidor de Salas (o de una Sala concreta), introducir su nombre de usuario y password (o introducir un nombre cualquiera si pueden acceder usuarios no registrados) e inmediatamente el sistema comienza a funcionar, transmitiendo audio y vídeo y visualizando al resto de participantes, permitiendo, de forma inmediata, la posibilidad de comunicarse con cualquiera de ellos. Además, su configuración es muy sencilla, ya que ofrece un número de opciones limitadas y fáciles de configurar. Por último, la utilización de las aplicaciones integradas también es muy sencilla de usar.
- Ideal para usuarios pocos experimentados con el PC y para equipos domésticos o portátiles. [9]

[9] Imagen cogida de `http://www.uwex.edu/disted/training/effect/james_byrnes.htm`

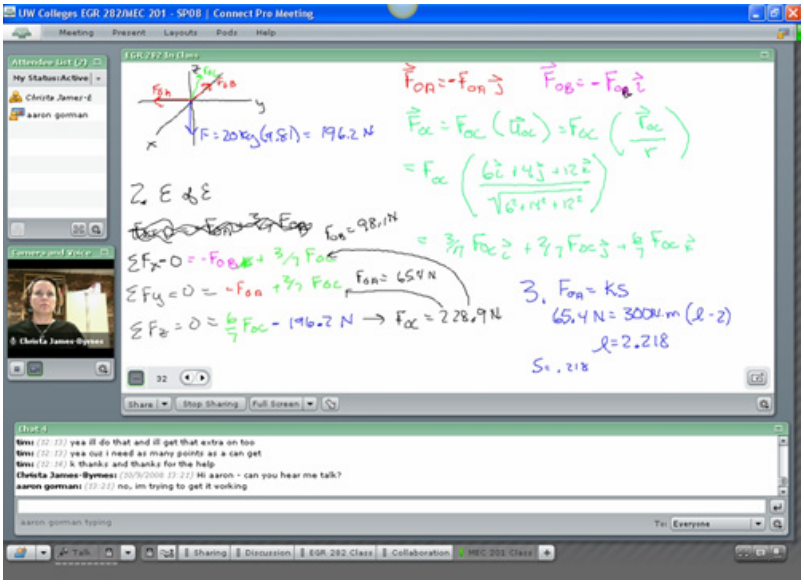


Figura 2.9: Usando la pizarra compartida de Adobe Connect

Sin embargo, *Adobe Connect* tiene una serie de desventajas de las cuales, podemos destacar las siguientes:

- El sistema no es libre ni gratuito, es un sistema propietario y cerrado, de pago, además de poseer un **alto coste de adquisición**. En este caso *Adobe* cobra una cuota mensual y su precio varía en función al número de usuarios máximos que se conecta. Por ejemplo, la versión más barata cuesta aproximadamente $39, que permite conectarse hasta 15 usuarios.

- Aunque el sistema puede utilizarse en cualquier sistema operativo (al tratarse de una aplicación basada en Flash y ejecutarse dentro de un navegador web), **no ofrece un buen rendimiento en sistema Linux**, ya que el Plugin del reproductor de Flash no está optimizado para estos sistemas. Por tanto, puede darse el caso de ralentizaciones y/o pixelaciones si se ejecuta *Adobe Connect* en estos sistemas.

- Como suele ser habitual, un sistema fácil de usar **limita las posibilidades de configuración y optimización** para los usuarios experimentados. Por ejemplo, la calidad de vídeo que ofrece *Adobe Connect* es aceptable, sin embargo, al ser tan fácil de utilizar, el sistema no ofrece ninguna configuración avanzada para optimizar la calidad de vídeo tanto para retransmitir, como para recibir. No permite cambiar el códec de vídeo a utilizar, ni la tasa de transferencia de datos (*bitrate*). Ocurre lo mismo con el audio, aunque ofrece un asistente para ofrecer el mejor audio posible, las opciones disponibles para su configuración es bastante limitada.

- Como se ha comentado anteriormente, este sistema es ideal para sistemas domésticos y/o portátiles. La razón principal es que todo el sistema **está integrado en una sola ventana**. Es decir, si tenemos, por ejemplo 5 ventanas de vídeo, un chat y, además, estamos compartiendo una presentación, todo ésto se visualiza en el interior de una ventana (a diferencia de otros sistemas donde cada ventana de vídeo y la presentación se muestran en ventanas independientes). Ésto supone un problema para grandes Salas (como las instaladas en la Universidad de Cádiz), ya que estas Salas ofrecen múltiples pantallas. Estas pantallas están conectadas al equipo y funcionan como un escritorio extendido, así que, lo ideal, es poder extender la ventana a todo el escritorio extendido. Sin embargo, ésto no es posible con *Adobe Connect*, únicamente nos posibilita ubicar la ventana del sistema en uno de los escritorios, limitando, así, el espacio disponible para visualizar la aplicación.

- A diferencia de otros sistemas de multiconferencia, *Adobe Connect* permite, únicamente, emitir una cámara por participante. En grandes Salas, como las que hay instaladas en la Universidad de Cádiz, es recomendable disponer de varias cámaras, para poder emitir el mayor número de detalles de dicha Sala. Por ejemplo, si se está realizando una clase a través de multiconferencia, sería ideal que, además de emitir por vídeo al profesor que está impartiendo la clase, poder emitir a los alumnos que están sentados en dicha Sala, por si desean preguntar alguna duda. Con *Adobe Connect* solamente podemos emitir una sola cámara, por lo que sería necesario otras soluciones audiovisuales que nos ayuden a realizar lo deseado, por ejemplo, una mesa de mezclas de vídeo, que nos permita cambiar entre varias cámaras la que queremos emitir en ese momento.
- Desincronización entre el audio y el vídeo. En muchas ocasiones, en una sesión de *Adobe Connect*, podemos observar cómo los participantes están moviendo los labios y el sonido que están emitiendo tarda en llegar, produciéndose una desincronización entre el audio y el vídeo.
- Como consecuencia de que *Adobe Connect* sea un sistema cerrado, no es posible mejorar ni ampliar el sistema añadiendo nuevas funcionalidades que se necesiten.

2.6. ¿Por qué Access Grid

Access Grid no es el sistema de colaboración multimedia perfecto, también tiene sus desventajas, de las cuales podemos destacar las siguientes:

- Sistema complejo de configurar. *Access Grid* es un sistema muy completo y muy versátil, nos permite añadir tantas cámaras como deseemos y cada una de ellas con un códec de vídeo diferente y una calidad de vídeo diferente, podemos configurar de igual manera el audio, permitiendo asignar un volumen adecuado a cada participante, etc. Sin embargo, configurar y averiguar los valores óptimos es bastante complejo para un usuario normal.
- Aunque, al igual que *Adobe Connect*, incorpora herramientas para gestionar usuarios y salas, dichas herramientas son más complejas de utilizar que *Adobe Connect.*
- Puede parecer, de primeras, un sistema inestable, sobretodo si el usuario comienza a configurar opciones de forma aleatoria y sin seguir unas pautas, a veces puede el sistema dar un error y cerrarse de forma inesperada. Sin embargo, una vez se conoce mejor al sistema y su comportamiento y, siguiendo unas pautas de configuración, el sistema es totalmente estable.
- La documentación existente es escasa (y prácticamente inexistente a nivel nacional). El sitio oficial de *Access Grid* [10] dispone de un *mailing list* tanto para usuarios como para desarrolladores, donde uno puede inscribirse y preguntar las dudas que tenga.

Sin embargo, las ventajas que posee *Access Grid* son numerosas y, por ese motivo, se ha escogido como sistema principal de teledocencia para las Salas instaladas en la Universidad de Cádiz. A continuación se detallan las principales ventajas:

Software Libre

Como con cualquier otro software libre, los usuarios tienen libertad total para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado. Es decir, pueden disponer de tantas copias como haga falta, sin tener que pagar nada por el software y, además, pueden modificarlo para adaptarlo a sus necesidades sin infringir ninguna ley. Ésto supone un ahorro considerable de costes.

[10] `http://www.accessgrid.org/`

Sistema Multiplataforma

Los desarrolladores de *Access Grid* ofrecen, para sus usuarios, versiones y soporte para los sistemas Mac, Linux y Windows, ofreciendo en todos un rendimiento similar, es decir, el usuario podrá decantarse por su versión de sistema preferido ya que el sistema *Access Grid* funciona igual en todos ellos. Además, el sistema funciona correctamente independientemente de en qué sistemas se está ejecutando *Access Grid* en el resto de participantes.

Sencillez de instalación y uso

Access Grid ofrece instaladores automatizados para cada uno de los Sistemas Operativos que soporta, facilitando su instalación en el sistema. Aunque, como se ha comentado, el sistema puede parecer complejo de configurar, una vez conseguido, se puede almacenar dicha configuración para que, cada vez que se ejecute el sistema, automáticamente se encuentre configurado y preparado para su uso, facilitando así la labor del usuario (únicamente debería de introducir la dirección del Servidor de Salas al que desea acceder).

Coste Escalable

Como se ha comentado anteriormente, el software es libre y, por tanto, gratuito, no hay gastos asociados a la copia del software por cada nueva instalación. Así, el coste de instalar este sistema se limita al hardware que se quiera usar con él. Puede optar por un sistema de bajo coste, utilizando un PC antiguo con una webcam y un micrófono; o bien, puede optar por un sistema de última tecnología en audio y vídeo instalados en una gran sala de conferencias.

Plugins de expansión

El sistema, además de ser software libre, es de **código abierto**. Así, una de las ventajas que tenemos es que podemos modificar a nuestro antojo el código fuente del sistema para corregir o mejorar funcionalidades de él. Además, dispone de una API para desarrollar nuevas aplicaciones integradas en *Access Grid*, permitiendo así, añadir nuevas funcionalidades. Se puede decir que el sistema es personalizable según nuestras necesidades.

Múltiples cámaras simultáneas desde un mismo cliente

A diferencia de los otros sistemas de colaboración que se han comentado anteriormente, *Access Grid* permite emitir más de una fuente de vídeo al resto de participantes, pudiendo enviar varias cámaras de vídeo que enfoquen lugares diferentes (ideal para grandes Salas).

Distribución libre de ventanas

Se comentó, como desventaja, en los otros sistemas de colaboración, que todo el sistema está integrado en una sola ventana. Ésto suponía una desventaja porque, en grandes salas, donde existen varias pantallas interconectadas formando un escritorio extendido, no permitían aprovechar todo el espacio que ello ofrece. En *Access Grid*, cada ventana de vídeo se muestra como una ventana independiente, pudiendo asignarle un tamaño y posicionarla a lo largo del escritorio extendido, permitiendo aprovechar mejor el espacio disponible.

Figura 2.10: Aprovechamiento del escritorio extendido con Access Grid

Calidad de audio

Lo fundamental en cualquier sistema de reunión remota es la calidad del audio. *Access Grid* dispone de un gran conjunto de opciones para configurar y optimizar la calidad del audio emitido y recibido. Además dispone de herramientas de diagnóstico de la recepción de los paquetes de datos de audio con el fin de informar sobre cualquier pérdida por mínima que ésta sea.

Sincronización del audio y del vídeo

Los desarrolladores se han esforzado en que las comunicaciones en las reuniones sean en tiempo real entre los participantes, lo cual hace que el vídeo y el audio de cada participante estén sincronizados, en el sentido de que cuando veamos al participante mover los labios, inmediatamente oiremos lo que está diciendo.

Multicast

Access Grid está totalmente preparado y orientado a dar servicio usando multicast. Ésto implica un óptimo aprovechamiento del ancho de banda y la posibilidad de poder emitir y recibir con un mayor bitrate (más alta calidad). El sistema no se resiente por tener un gran número de participantes.

Apuesta de futuro

Este software goza de un gran apoyo tanto por parte la comunidad opensource como la comunidad universitaria occidental. Los cuales, en un mundo cada vez más acostumbrado al uso de la red, buscan soluciones con las que realizar reuniones que se asemejen lo más posible a una reunión física pero sin los costes en tiempo y dinero de los desplazamientos necesarios para ellas.

Capítulo 3

Planificación del proyecto

3.1. Organización temporal

3.1.1. Planificación Inicial

En este apartado hablaremos sobre la planificación temporal que se realizó para nuestro proyecto. Debemos tener en cuenta que este proyecto se trataba de un proyecto novedoso, tanto para la empresa Auditel, como para la Universidad de Cádiz, ya que ninguna de las dos partes habían trabajado nunca en un proyecto similar. Auditel es una empresa cuyos proyectos son instalaciones a gran escala: instalación del sistema de seguridad de un estadio, instalaciones eléctricas y mecánicas en diversos edificios de la empresa Airbus, etc (para más detalle: `http://www.auditel.es/referencias.htm`), pero realmente nunca antes habían instalado una sala de teledocencia, ni habían trabajado con *Access Grid*.

Por tanto, hay que destacar que se planificaron algunos objetivos, pero otros, por entonces, aún no estaban muy claros, sino que, dependiendo de los resultados que fuéramos obteniendo, se hacían o no, o incluso podían surgir nuevos objetivos. Consecuentemente, tampoco se sabía a ciencia cierta cuánto iba a durar el desarrollo de cada uno de los objetivos, se hizo unas estimaciones pero sin tener la certeza de que, realmente, ese tiempo planificado iba a ser suficiente, o sobraba, para cada objetivo.

Sin embargo, desde el comienzo del proyecto, existían ciertos objetivos en el que todos estaban de acuerdo que habían que cumplirse de una forma u otra. Éstos son:

- **Configurar y optimizar el sistema *Access Grid*:** Se podría decir que era lo que más interés tenía todo el mundo que se cumpliera. Nadie había trabajado (ni si quiera visto) nunca con el sistema *Access Grid*, de primeras puede parecer complejo de utilizar y configurar, además de que era inestable y surgían errores inexplicables para ellos.
- **Migrar el Servidor de Salas instalado en Cádiz a Puerto Real:** El Servidor de Salas de *Access Grid* se instaló en el Campus de Cádiz, en el Aulario La Bomba. Ésto tenía el inconveniente de que dicho servidor no estaba siempre encendido, y había que ponerse en contacto con el personal de Cádiz cada vez que era necesario conectarse a una sesión de *Access Grid* utilizando nuestro Servidor de Salas. Era necesario trasladar dicho Servidor de Salas a la Sala de Máquinas del Campus de Puerto Real, donde, además de recibir un mantenimiento por parte de los Administradores de Sistemas de allí, la máquina estaría siempre disponible cada vez que lo necesitáramos.
- **Migrar el Sistema *Access Grid* a la Sala de Jerez:** Desde hace unos años, existe una Sala de Teledocencia en el Campus de Jerez, donde se realizaban videoconferencias a través del sistema

Polycom. Esta sala ha quedado algo desfasada en cuanto a equipamiento y, en general, a tecnología tanto hardware como software. Se acordó en que Auditel actualizará dicho equipamiento a esta Sala y tendríamos que migrar el sistema *Access Grid* a dicha Sala.

- **Presupuesto mínimo:** Se trata de investigar el equipamiento mínimo necesario y, por tanto, establecer un presupuesto mínimo para poder instalar una Sala de Teledocencia aceptable.
- **Desarrollar una aplicación para poder visualizar presentaciones y difundirlas:** Una de las carencias que notaron en la Universidad de Cádiz era que, a la hora de realizar algún evento a través de Access Grid, si querían mostrar una presentación (en PowerPoint, OpenOffice o PDF) o bien difundían la presentación a todos los participantes (vía email, etc) y luego avisaban cada vez que había que cambiar de transparencia, o bien, a través del software *VLC*, se capturaba el escritorio y se enviaba a la red. Sin embargo, este método no era muy efectivo, ya que, aunque se viera el escritorio, lo que realmente se envía es vídeo, perdiéndose así calidad de imagen, por lo que no se llegaba a ver algunas presentaciones. Además, el sistema *Access Grid* es algo inestable y han surgido errores inesperados. Como *Access Grid* es software libre y de código abierto, otro objetivo sería corregir el código fuente del sistema.
- **Restringir el acceso a una Sala:** Aunque la filosofía de Access Grid es libre, es necesario disponer de un control de acceso, ya que, dentro de un Servidor de Salas, pueden crearse varias Salas. Sería interesante poder restringir el acceso para aquellas Salas que se consideren privadas, por ejemplo, tener una Sala para reuniones de Vicerrectores, no sería aceptable que cualquiera pudiese entrar en dicha reunión.

Así pues, entre varias reuniones que se tuvo al comienzo del proyecto, la planificación temporal que resultó es la siguiente:

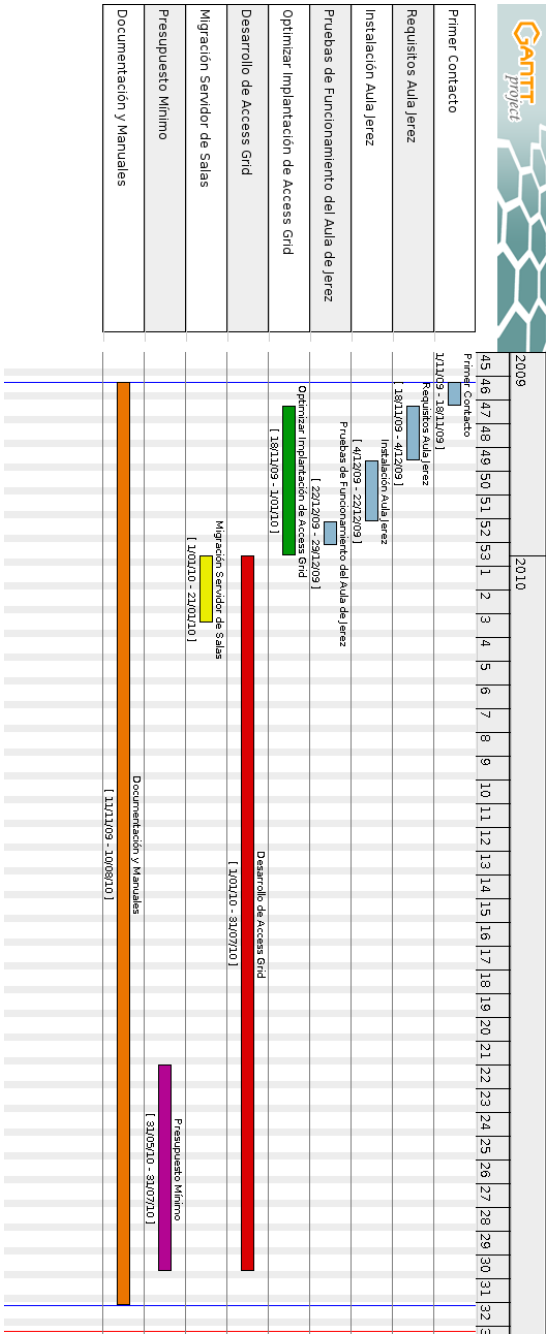


Figura 3.1: Planificación Temporal Inicial del Proyecto

A continuación se detallan cada una de las fases del proyecto:

- **Primer Contacto:** 11 de Noviembre de 2009 hasta 18 de Noviembre de 2009. Se trata de una fase de iniciación, ya que ni Miguel ni yo habíamos utilizado nunca el sistema *Access Grid*. Se trataba de conocer un poco el sistema, ver sus funcionalidades, etc.

- **Instalación Aula de Jerez:** Se trata del conjunto de fases en las que se establecen los requisitos necesarios para el Servidor de Jerez, su instalación y una fase de pruebas para verificar su correcto

funcionamiento. La idea es que, antes de que empezara el año nuevo, estuviera la Sala de Jerez completamente funcional. Estas fases son :

- **Requisitos Aula Jerez:** 18 de Noviembre de 2009 hasta 4 de Diciembre de 2009. Se trata de una fase en la que, teniendo en cuenta el equipamiento hardware instalado, y el equipamiento que la empresa Auditel va a actualizar y añadir, establecer los requisitos necesarios para el equipo que se instalará en dicha aula.
- **Instalación Aula Jerez:** 4 de Diciembre de 2009 hasta 22 de Diciembre de 2009. Se trata de una fase en la que comenzarían las obras de renovación del aula de Jerez, por parte de la empresa de Auditel y, una vez finalizado, trasladar e instalar el equipo adquirido en dicha aula.
- **Pruebas de Funcionamiento del Aula de Jerez:** 22 de Diciembre de 2009 hasta 29 de Diciembre de 2009. Se trata de una fase de pruebas para verificar el correcto funcionamiento del equipamiento instalado en la Sala, así como del equipo adquirido.

- **Migración del Servidor de Salas**: 1 de Enero de 2010 hasta 21 de Enero de 2010. Esta fase comprende tanto la adquisición del Servidor, instalarlo en la Sala de Máquinas del Campus de Puerto Real, instalación del Sistema Operativo y software necesario, así como la migración del software del Servidor de Salas del Campus de Cádiz.

- **Optimizar la implantación de Access Grid:** 18 de Noviembre de 2009 hasta 1 de Enero de 2010. Esta fase corresponde al conjunto de tareas a realizar para poder optimizar la configuración del sistema *Access Grid* obteniendo, así, la mejor calidad y el mejor rendimiento del sistema.

- **Presupuesto mínimo:** 31 de Mayo de 2010 hasta 31 de Julio de 2010. En esta fase se investigará el equipamiento mínimo para poder instalar un aula de Teledocencia de forma aceptable, obteniendo el mínimo coste.

- **Desarrollo de Access Grid:** 1 de Enero de 2010 hasta 31 de Julio de 2010. En esta fase se desarrollará aplicaciones de mejora para Access Grid, principalmente una aplicación para poder visualizar presentaciones y difundirlas al resto de participantes, además de la corrección del código fuente para mejorar la estabilidad del sistema. Por último, se investigará sobre cómo restringir el acceso a una Sala de Access Grid.

- **Documentación y Manuales:** 11 de Noviembre de 2009 hasta 10 de Agosto de 2010. Durante la realización de este proyecto se ha ido documentando todo lo relacionado con las Salas de Teledocencia. Además, conforme se iba avanzando en el desarrollo del proyecto, también se fue elaborando la documentación de este Proyecto.

3.1.2. Planificación Final

Como se ha comentado anteriormente, tanto la Universidad de Cádiz, como la empresa Auditel, se enfrentaban por primera vez a un proyecto como éste y, por tanto, la planificación no estaba del todo clara y la duración de cada una de las tareas tampoco. Además, han surgido varios problemas en los que, debido a éstos, se han retrasado aún más la realización de cada una de las tareas. Algunos de estos problemas son los siguientes (se detallarán mejor en próximos capítulos):

- Nuestra fase de iniciación con el sistema *Access Grid* necesitó más tiempo ya que, entre otros motivos, muchas veces era necesario establecer una conexión con una Sala y nuestro Servidor de Salas del Campus de Cádiz estaba apagado. Fue necesario crearnos un marco de trabajo, donde

podíamos instalar nuestro propio servidor de salas de pruebas y dos clientes para conectarse a dicho servidor y realizar las pruebas que veíamos necesarias.

- Surgieron varios problemas para la Sala de Jerez. Por un lado, se retrasó la entrega del equipo que solicitamos y, cuando llegó, faltaban aún algunos componentes por llegar y nos dieron otros diferentes mientras esperábamos los que habíamos solicitado. Por otro lado, el equipo vino con problemas de congelamiento e inestabilidad sin saber el motivo. Por último, nos encontramos con un error de instalación en la cancelación del eco, provocando, consecuentemente, que hubiera eco en las sesiones tanto de *Access Grid* como de *Adobe Connect*.
- La optimización de la implantación de Access Grid se retrasó considerablemente, principalmente, debido a un problema de pixelación de vídeo.
- La migración del Servidor de Salas a la Sala de Máquinas del Campus de Puerto Real se hizo más tarde, debido a que el equipo que se solicitó tardaba más de lo que se había planificado.
- Consecuentemente a los retrasos anteriores, la fase de Desarrollo de Access Grid tuvo que iniciarse más tarde de lo planificado.

Como se ha comentado, todos estos problemas se hablarán con más detalles en próximos capítulos de esta documentación. Así pues, la planificación real de este proyecto es la siguiente:

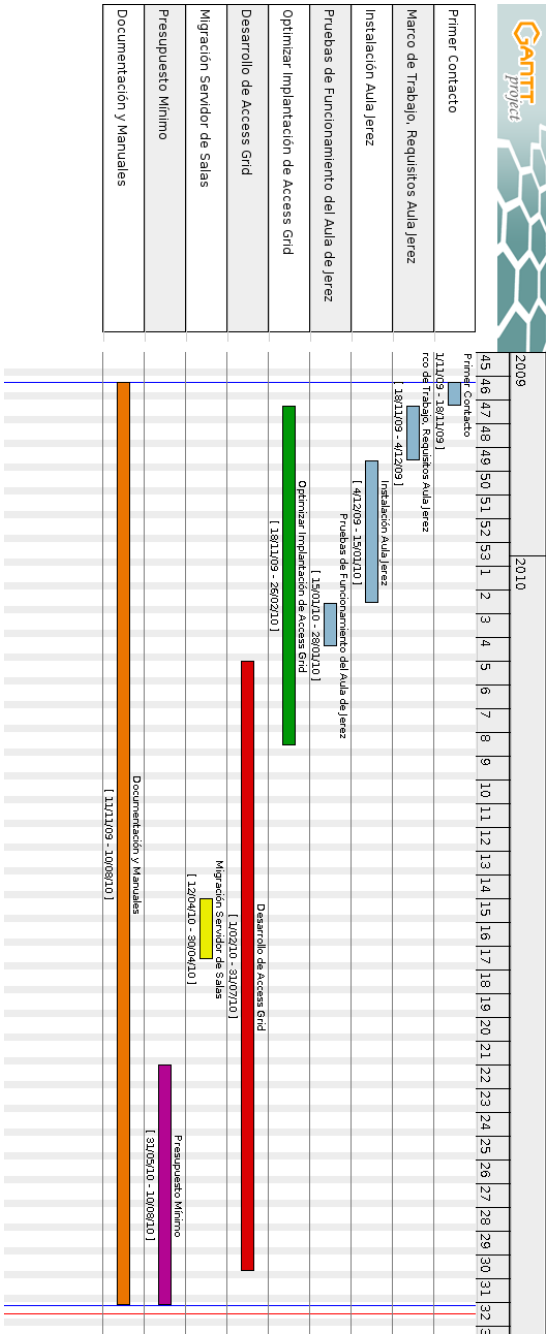


Figura 3.2: Planificación Temporal Final del Proyecto

Capítulo 4

Fase de Implantación

4.1. Organización

Para explicar lo mejor posible cómo se ha implantado el sistema *Access Grid*, así como su uso, configuración y optimización, se ha dividido este capítulo en varios bloques. A continuación, se detallan estos bloques:

- **Uso como usuario final/operador:** En este bloque se explicará todo el proceso de aprendizaje que se llevó a cabo para utilizar y configurar el sistema *Access Grid*
  - **Punto de Partida y Características de las Salas:** Aquí, se explicará el comienzo del proyecto, qué había ya hecho y cuáles son las características de las Salas de Teledocencia, detallando su equipamiento instalado.
  - **Instalación de nuestro Sistema *Access Grid*:** Se explicará cómo se creo un sistema *Access Grid* para poder realizar pruebas y aprender a utilizarlo.
  - **Instalación de los Clientes *Access Grid*:** Aquí se explicará todo el proceso de aprendizaje seguido para saber utilizar y configurar el cliente de *Access Grid*, detallando cada una de sus opciones, etc.
  - **Configuración del *Servidor de Salas*:** Por otro lado, se explicará todo el proceso de aprendizaje seguido para saber utilizar y configurar el *Servidor de Salas*.
- **Puesta a punto del Sistema:** En este bloque se explicará todo el proceso de instalación, configuración y optimización de las Salas de Teledocencia, poniendo en práctica el conocimiento adquirido en el apartado anterior.
  - **Configuración de los Clientes:** Se explicará la configuración aplicada a los clientes de *Access Grid* de las Salas de Teledocencia.
  - **Problemas encontrados:** Se hablará de todos los problemas que han surgido a lo largo de este proyecto, desde pixelaciones de vídeo, problemas de eco hasta problemas de redes.
  - **Uso y configuración del equipamiento de la Sala:** Se hablará sobre el aprendizaje para utilizar todo el equipamiento de la Sala.
  - **Migración del Servidor de Salas:** Aquí se contará todas las acciones que se han llevado a cabo para migrar el Servidor del Campus de Cádiz al Campus de Puerto Real. Además de cómo se ha creado un Bridge multicast, así como el Control de Acceso y la Gestión de Salas
  - **Copias de seguridad:** Se hablará sobre las políticas utilizadas para realizar copias de seguridad de los sistemas de las Salas de Teledocencia.

- **Instalación del Aula de Jerez:** En este bloque se hablará sobre todo el proceso de instalación y puesta a punto de la Sala de Teledocencia instalada en el Campus de Jerez, así como los problemas encontrados y las soluciones aportadas.

- **Certificación de las Salas de Teledocencia:** En este bloque se hablará sobre la Certificación conseguida de cada una de las Salas de Teledocencia, garantizando, con ella, que las Salas están capacitadas para utilizar el sistema *Access Grid*.

- **Presupuesto mínimo:** En este bloque se hablará sobre qué componentes, como mínimo, son necesarios para poder utilizar el sistema *Access Grid* en un aula o un despacho, así como el coste asociado por ello.

4.2. Wiki de *Access Grid*

Si el lector lo desea, hay disponible una *wiki* sobre *Access Grid*, que se ha ido utilizando a lo largo de este Proyecto. En el se detallan todos los errores encontrados, así como las soluciones aportadas. Gracias a ésto se ha podido generar un manual completo de *Access Grid*, que también está disponible como Apéndice en esta documentación.

La dirección de la wiki es la siguiente: `http://softwarelibre.uca.es:9000/redmine/wiki/teledocencia`.

4.3. Uso como usuario final/operador

En este capítulo se detallará todos los acontecimientos que han surgido a lo largo de este proyecto donde, partiendo sin tener conocimiento alguno sobre el sistema *Access Grid* y, en general, sobre el equipamiento instalado en las Salas de Teledocencia, se consiguió tener una buena formación y se obtuvo los conocimientos necesarios tanto para el uso del sistema *Access Grid* como de la propia Sala.

4.3.1. Punto de Partida y Características de las Salas

A principios de Octubre de 2009 fuimos convocados para una reunión para comunicarnos en qué consistiría nuestra labor. Una vez reunidos, nos enseñaron la Sala de Teledocencia del Campus de Puerto Real y, al entrar en la cabina, nos encontramos con que debíamos de saber manejar un conjunto de equipos de alta tecnología que nunca antes habíamos visto.

Figura 4.1: Interior de la Cabina I

Figura 4.2: Interior de la Cabina II

Características de las Salas

A continuación, se describirán las características de las Salas de Teledocencia instaladas en la Universidad de Cádiz. Estas Salas han sido diseñadas con vistas de futuras ampliaciones. Una Sala se compone de tres habitáculos:

- Un Aula: Donde se da lugar a todos los acontecimientos que se pueden dar en la Sala de Teledocencia (impartir clases, reuniones, congresos, etc.).
- Una cabina de control: Donde se encuentra todo el equipamiento de la Sala y su control.
- Una cabina de traducción: Como futura ampliación (actualmente no hay nada instalado), se ha creado una cabina de traducción para aquellas sesiones donde sea necesario una traducción simultánea.

La ubicación del equipamiento objeto del equipamiento de control se encuentra en la cabina de control. Todos los equipos instalados en el interior del aula, altavoces, cámaras, micrófonos, etc., son gestionados a través de equipos situados en dicha cabina. El operador de la Sala puede gestionar ciertos equipos de forma manual y remota a través de un sistema de control remoto AMX.

Figura 4.3: Control Remoto del Equipamiento por AMX

Como hemos comentado anteriormente, las Salas de Teledocencia disponen de dos cabinas, uno destinado a ubicación de equipos y el otro a cuarto de traductores, como futura ampliación. En el cuarto de traductores dispone de la canalización necesaria para, en un futuro, instalar pupitres de intérprete.

En cuanto al equipamiento de la Sala disponemos de lo siguiente:

Equipamiento Audiovisual

- Microfonía inalámbrica formada por un micrófono de mano y un micrófono de solapa o de petaca.
- Microfonía de debate formada por una unidad de micrófono de Presidencia y 17 unidades de micrófonos de delegados.
- Sistema de tratamiento digital del sonido, para evitar la realimentación del sonido (eco).
- Mesa de mezclas de audio digital donde se conecta toda la microfonía, además de otro equipamiento (reproductor de DVD, PCs, etc.)
- Procesador profesional de sonido Dolby Digital 7.1, para la reproducción del sonido digital.
- Conjunto de altavoces instalados en la Sala y en la cabina de control para monitorizar el sonido emitido y recibido.
- Matrices de vídeo y componentes RGBHV para enrutar las diferentes fuentes de vídeo y de sonido.

Equipamiento de Imagen

- Tres videoproyectores para la videoproyección múltiple.
- Pantalla fija con superficie de visualización 609x152 cm., lo que equivale a tres pantallas de 100″ colocadas una al lado de la otra.
- Tres cámaras motorizadas.

- Mezclador de vídeo.
- Dos reproductores/grabadores de DVD con disco duro integrado.
- Sistemas de codificación y decodificación MPEG-2.
- Tarjetas capturadoras de vídeo.
- Conversor VGA-PAL, para poder inyectar señales de VGA en sistema de vídeo compuesto instalado.
- Matriz de vídeo compuesto.
- Monitores previo y de programa, para monitorizar el vídeo que se está retransmitiendo y/o enviando.
- Monitores LCD instalados en la mesa de presidencia que sirvan como referencia visual a lo que el público está visualizando en la pantalla.

Equipamiento para el control de todos los sistemas audiovisuales

- Toda la gestión de los medios audiovisuales antes definidos se realiza a través de un panel táctil inalámbrico de 7" y de unas botoneras preconfigurables.
- El sistema de iluminación, además de poder controlarse por AMX, también se puede controlar a través de una botonera instalada en la Sala.

Equipamiento Informático

- PC cuya función es la de Servidor de Visualización y Salas, tanto para *Access Grid* como para *Adobe Connect*, bajo Windows XP.
- PC cuya función es la de Servidor de Vídeo y de Audio para *Access Grid*, bajo Red Hat.
- PC cuya función es la de Codificador de Windows Media, bajo Windows XP.

Equipamiento para Seguridad y Control de Accesos

- Vigilancia por circuito cerrado de TV.
- Sistema de grabación por disco duro, para almacenar las grabaciones del sistema de vigilancia.
- Lector de tarjetas para el control de acceso a las cabinas de control y de traducción.
- Sistema de intrusión, formado por una central de intrusión, una sirena y un detector volumétrico situado en el interior del cuarto de control.

Como se puede observar, una Sala de Teledocencia dispone de un gran número de equipos dotados de la última tecnología. Es necesario tener una formación para poder utilizar cada equipo, además de poder sacar el máximo provecho a cada uno de ellos.

4.3.2. Instalación de nuestro sistema Access Grid

Comenzamos nuestra fase de preparación intentando conocer el sistema *Access Grid*, un sistema totalmente desconocido para nosotros. Lo primero que hicimos fue intentar instalar el sistema tanto en Windows XP como en Ubuntu. Tanto en un sistema como en otro, no tuvimos ningún problema para instalar *Access Grid*. Después de algunas pruebas vimos que al instalar el software *Access Grid* se instala tanto el Cliente como el Servidor de Salas, dependiendo del uso que queramos dar a un equipo determinado, ejecutaremos una u otra aplicación.

Creamos nuestro propio marco de trabajo, donde instalaríamos un Servidor de Salas de pruebas y dos clientes de *Access Grid* para conectarnos a dicho servidor.

En el Servidor de Salas, hemos optado por instalar, como sistema operativo, el sistema **Ubuntu Server 9.10**. Los motivos de haber elegido este sistema son varios. Por un lado, el Servidor de Salas que está instalado en el Campus de Cádiz está instalado bajo el sistema Windows XP. Como uno de nuestros objetivos es migrar dicho servidor al Campus de Puerto Real, queríamos saber si existe alguna diferencia en instalar el Servidor de Salas en Windows o en Linux. Además, como uno de los PCs de la Sala utiliza Red Hat, fue otro motivo más para optar por Ubuntu, para ver si también existen diferencias entre las diferentes distribuciones de Linux. Por último, a la hora de documentar todo lo relacionado con este sistema, sería conveniente saber utilizarlo tanto en Windows como en Linux.

En cuanto a los clientes, están instalados tanto para Ubuntu, como para Windows (tanto Windows XP como Windows 7). Los motivos de su instalación en los diferentes sistemas son los mismos comentados en el párrafo anterior.

Una vez instalado el sistema operativo y el Servidor de Salas de *Access Grid*, procedemos a ejecutar los programas necesario para establecer una sesión. Es necesario tener instalado un certificado digital cuyos datos estén asociados tanto a la máquina, como a *Access Grid*.

Para ello, hay que solicitar un certificado digital, a través de una aplicación integrada en el propio *Access Grid*. Se solicitará una serie de datos, entre ellos, el **servidor de nombres de la máquina** (el DNS). Tuvimos que ponernos en contacto con el departamento de redes del C.I.T.I. para solicitarles un DNS para nuestra máquina. El nombre de la máquina es `agserver.uca.es`.

4.3.3. Instalación de los Clientes Access Grid

Como se ha comentado anteriormente, conseguimos instalar el cliente *Access Grid* tanto en Ubuntu, como en Windows. En cualquier sistema donde instalemos *Access Grid* nos aparecerá la pantalla principal:

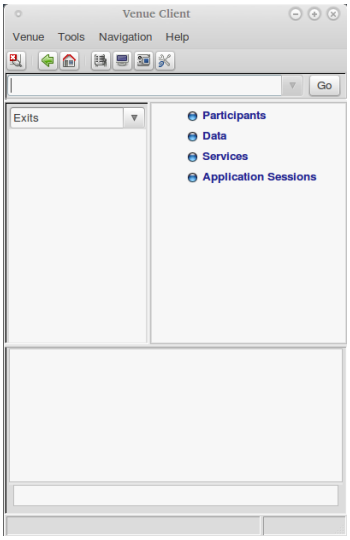


Figura 4.4: Pantalla Principal de Access Grid

Así pues, se pasó a aprender el funcionamiento de este sistema.

Servicios en Access Grid

Access Grid, entre otras funciones, gestiona la recepción y emisión tanto de sonido como de vídeo a través de **servicios**. Los servicios son, básicamente, contenedores de software que pueden ejecutarse en un Cliente de *Access Grid* (*VenueClient*) cuando dicho Cliente accede a una Sala. Los servicios pueden ser de diversos tipos, incluso, si se desea, se pueden diseñar servicios personalizados, empaquetarlos, distribuirlos y cargarlos en el Cliente para añadir nuevas funcionalidades.

Los servicios más usuales son:

- **AudioService:** Para emitir y recibir el sonido a través de la aplicación RAT.
- **VideoService:** Para emitir y recibir el vídeo a través de la aplicación Vic.
- **VideoConsumerService:** Para únicamente recibir el vídeo (sin emitir). Opción que deben utilizar aquellos PCs que no dispongan de WebCam.
- **VideoProducerService:** Para únicamente emitir el vídeo (sin recibir).
- **Otros Servicios:** Dependiendo de los paquetes que hayamos instalado, pueden aparecer nuevos servicios, como por ejemplo VideoServiceH264 para la retransmisión y emisión de vídeo utilizando el códec H.264, o HDVideoService para la retransmisión y emisión de vídeo en HD.

Por otro lado, se observó que *Access Grid* se apoya principalmente en dos programas: ***Vic*** (*Video Conference Tool*), cuya funcionalidad es gestionar todo el flujo de vídeo tanto de entrada como de salida, y ***RAT*** (*Robust Audio Tool*, que gestiona el flujo de audio tanto de entrada como de salida. Estas aplicaciones se invocan al añadir algún servicio de audio o vídeo en nuestro Cliente *Access Grid*.

Es necesario configurar el cliente de *Access Grid* para adaptarlo a nuestro hardware, tanto el audio como el vídeo.

Cómo añadir servicios

En el manual completo de *Access Grid*, o en la wiki mencionada anteriormente, se detalla los pasos a seguir para añadir servicios a *Access Grid*.

Por otro lado, en un principio se pensaba que, tanto en Windows como en Linux, se ofrecían los mismos servicios. Sin embargo, se comprobó que ésto no era así, ya que, durante la instalación del sistema *Access Grid*, en los sistemas Windows, se nos pregunta qué servicios adicionales se desea instalar en el equipo.

Configuración de un servicio de Access Grid

Una vez aprendido cómo añadir servicios en *Access Grid*, el siguiente paso era saber configurarlos y, por último, optimizarlos.

Este proceso **llevó mucho más tiempo de lo planificado**, principalmente, por una serie de problemas de pixelación que en próximos capítulos se detallará más profundamente. Sin embargo, no era éste el único problema encontrado.

Al configurar el vídeo, una posible pantalla de configuración es la siguiente:

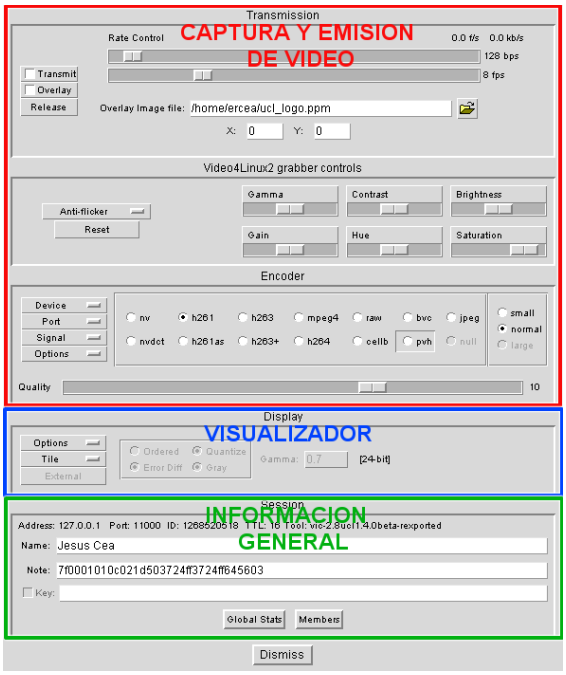


Figura 4.5: Opciones de configuración de vídeo en Access Grid

Como se puede observar en la imagen, el menú de configuración de Vic está dividido en tres grandes grupos. El primero de ellos es el conjunto de opciones para la retransmisión de vídeo, el segundo grupo es para la recepción de vídeo y el tercer grupo es simplemente una serie de información general. Así pues, dependiendo del tipo de servicio de vídeo agregado (emisión de vídeo o captura de vídeo) habrá que centrarse en un grupo u otro (o en ambos).

En el manual completo de *Access Grid*, o en la wiki mencionada anteriormente, se ha redactado un apartado muy completo donde se detalla cada una de las opciones disponibles tanto para configurar la aplicación *Vic*, para el vídeo, como para la aplicación *RAT*, para el audio.

En esta etapa se encontraron varios problemas. En primer lugar, en los clientes Linux, la webcam que disponíamos en nuestros puestos de trabajo no se visualizaba correctamente, mostrando una imagen distorsionada. Al principio se pensaba que era una mala configuración del servicio añadido, o incluso, una incompatibilidad del códec seleccionado, incluso se pensó en que podría ser un problema de drivers de la webcam en sistemas Linux.

Al final resultó ser una **incompatibilidad de la webcam utilizada con *Access Grid***. Este sistema, al estar en continuo desarrollo, no es compatible con todo el hardware existente, y tampoco en todos los sistemas operativos, por lo que **hay que tener cuidado con el hardware elegido para que forme parte de él**. En la página oficial, existe una sección llamada ”Hardware”, donde aparece un listado de todo el hardware testeado, así como los resultados obtenidos. Puede ser de utilidad para saber qué dispositivos han sido probados y saber si funcionan o no en *Access Grid*.

Por último, otro problema encontrado fue que, dependiendo del códec de vídeo utilizado, puede que tanto nosotros mismos, como el resto de Clientes, **no puedan visualizar una fuente de vídeo**. Dependiendo del códec utilizado para emitir vídeo, **es necesario añadir un servicio de recepción de vídeo del mismo tipo (códec)**. Es decir, si queremos emitir vídeo en H.264 ó MPEG-4, para que el resto de clientes puedan visualizarlo, ellos deberán añadir un servicio *VideoConsumerH264*.

Añadir servicios alojados en otra máquina

A la hora de añadir servicios a nuestro cliente de *Access Grid*, se observó una opción que no se sabía para que servía, los **ServiceManagers**.

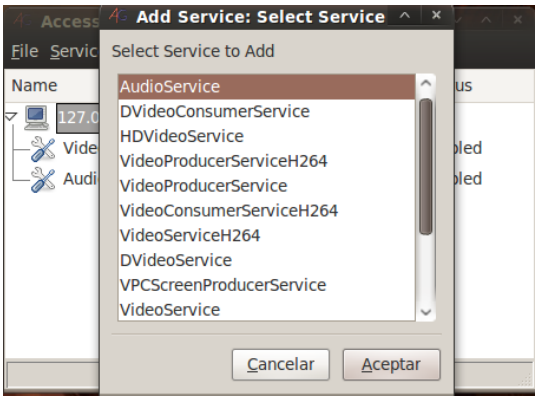


Figura 4.6: Listado de servicios a añadir en Access Grid

Hay que recordar que, en las Salas de Teledocencia, hay tres equipos interconectados. El primero de ellos, bajo el sistema *RedHat*, gestiona el vídeo a emitir y el audio. El segundo de ellos, el llamado ”Visualizador”, ejecuta el Cliente de *Access Grid* y recibe todo el flujo de vídeo. El último equipo es para utilizar el *Codificador de Windows Media*.

El sistema *Access Grid*, **permite añadir servicios que están alojados en otras máquinas**, de ahí que, desde Windows se pudiera acceder al hardware de la máquina de Red Hat. Para ello hay que seguir

un procedimiento a seguir:

- En el equipo donde se quiera poner a disposición sus servicios, hay que ejecutar la aplicación llamada ***ServiceManager***. Esta aplicación se ejecuta, bien en segundo plano, o bien, con una ventana de depuración. Lo que hace es **detectar todo el equipamiento**, tanto de audio y vídeo, que dispone el equipo y lo pone a disposición de la red, en busca de que un Cliente de *Access Grid* haga uso de sus recursos
- En el Cliente de *Access Grid* que quiera hacer uso de los recursos de la otra máquina, se debe añadir un **ServiceManager**, introduciendo la dirección IP de la máquina anterior.

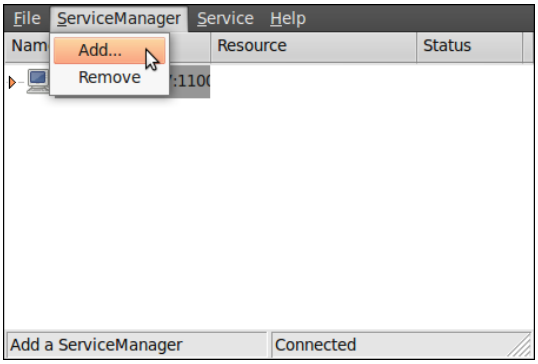


Figura 4.7: Añadir un ServiceManager

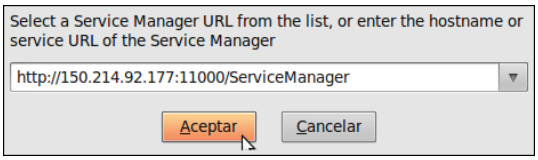


Figura 4.8: Introducir la dirección IP de la máquina

- Por último, se añadirían los servicios que se desean utilizar de la máquina remota, siguiendo el procedimiento habitual.

Configuración y valores óptimos de audio y vídeo

El largo aprendizaje seguido para configurar adecuadamente los servicios del sistema *Access Grid* permitieron adquirir un gran conocimiento acerca de las aplicaciones *Vic* y *RAT*, consiguiendo, así, saber, por un lado, cómo configurar cada uno de los programas adecuadamente y, sobre todo, **adaptar el software al hardware equipado**. Por otro lado, este conocimiento, también permite **solucionar problemas** que puedan surgir durante el uso del sistema *Access Grid*, incluso si dichos problemas nunca antes se habían producido.

Todo este conocimiento, permitió documentar un amplio apartado, en el manual completo de *Access Grid*, que lo puede encontrar en los Apéndices de esta documentación, o en la wiki anteriormente comentada, sobre cómo configurar, en detalle, el audio y el vídeo del sistema *Access Grid*, desde los valores más básicos, pasando por una configuración avanzada de cada uno de los parámetros disponibles tanto para la aplicación *Vic*, como para la aplicación *RAT*.

Cómo guardar la configuración de Access Grid

Por último, una vez configurado el Cliente de *Access Grid*, hay que **almacenar la configuración**, para no tener que repetir todo el proceso de configuración. Se pueden almacenar tantas configuraciones como se desee, pudiendo indicar, una de ellas, que sea la configuración por defecto. Así, cada vez que se ejecute el sistema *Access Grid*, se cargará dicha configuración sin que el usuario tenga que indicarlo.

Existen dos formas de almacenar la configuración de *Access Grid*. La más fácil y directa (y la más usual) es a través del propio Cliente de *Access Grid*. Sin embargo, una segunda forma de almacenar nuestra configuración, como se ha comentado, es **creándonos un fichero de configuración**.

Así pues, en el manual completo de *Access Grid*, que se adjunta en los Apéndices de esta documentación, o en la wiki anteriormente comentada, se ha añadido un capítulo extenso sobre cómo crear nuestro fichero de configuración, explicando qué parámetros existen y para qué sirven cada uno de ellos.

Conclusiones

Configurar el cliente *Access Grid* puede ser, a primera vista, **una tarea tediosa**. En primer lugar, es necesario saber si el hardware disponible es compatible con el software, según en qué Sistema Operativo se esté ejecutando. En la página oficial de *Access Grid*, existe un listado de hardware que ha sido testeado así, como el resultado obtenido (si existen problemas o no). [1]. Por otro lado, hay que añadir tantos servicios como elementos de audio y vídeo se quieran emitir y recibir. Configurarlos también puede parecer complicado.

Sin embargo, *Access Grid* **permite almacenar la configuración establecida**, así, una vez pasada esta, aparentemente, tediosa tarea de configuración, se puede almacenar para no tener que repetir todo el procedimiento.

Tanto los servicios de vídeo, como los servicios de audio, **permiten elegir un códec según nuestras necesidades**. Para los servicios de vídeo, los más destacados son los códecs ***H.261***, ***H.261as***, que permite emitir a una mayor resolución con el códec ***H.261*** y el códec ***MPEG-4***, ofreciendo la mejor calidad de vídeo, a costa de un consumo mayor de recursos. En cuanto al audio, por defecto emite a la mayor calidad posible, ya que, apenas consume recursos. Permite elegir un códec de un listado enorme para equipos antiguos que tengan problemas de rendimiento con el audio.

Por otro lado, **se pueden añadir servicios alojados en otra máquina**, permitiendo, así, repartir la carga de trabajo en varios equipos, mejorando el rendimiento del sistema.

4.3.4. Configuración del Servidor de Salas

Una vez aprendido el uso y configuración del Cliente de *Access Grid*, el siguiente objetivo fue saber usar y configurar el Servidor de Salas. Hasta ahora, únicamente se consiguió ejecutar la aplicación *VenueServer* e instalar un certificado para su correcto funcionamiento. Sin embargo, esta aplicación ofrece muchas más opciones y posibilidades.

Instalación del Certificado

Uno de los objetivos más fáciles fue conseguir instalar un certificado digital en nuestro Servidor de Salas. Ya se ha comentado el procedimiento a seguir en secciones anteriores.

[1] `http://www.accessgrid.org/hardware`

Sin embargo, *Access Grid* ofrece mucho más acerca de los certificados digitales. Dispone de una aplicación que se llama *Certificate Manager*:

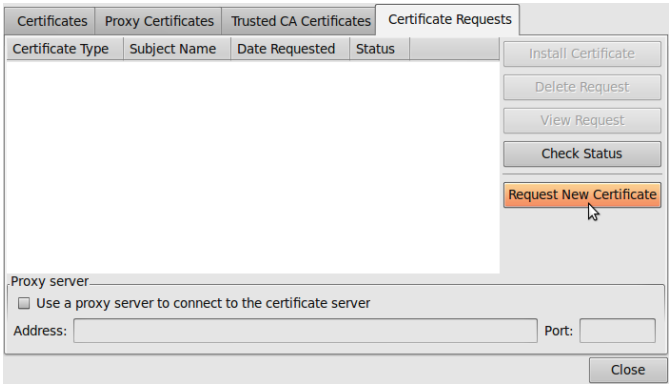


Figura 4.9: Gestor de Certificados de Access Grid

Desde esta aplicación, se puede solicitar un nuevo certificado digital, que así fue como se hizo para nuestro servidor de pruebas. Sin embargo, tanto la propia aplicación *Certificate Manager*, como la herramienta para solicitar certificados, tienen otras funcionalidades interesantes.

En primer lugar, *Certificate Manager* es una aplicación que **gestiona los certificados digitales** que utiliza el sistema *Access Grid*. Vemos que existen cuatro pestañas: *Certificates, Proxy Certificates, Trusted CA Certificates y Certificate Request*.

La pestaña *Certificate* muestra un listado de los Certificados actualmente instalados en el sistema *Access Grid*. La pestaña *Request Certificate* es donde fuimos para solicitar nuestro certificado digital.

En cuanto a la solicitud de certificados, recordemos que uno de los pasos nos preguntaba qué tipo de certificado queríamos instalar:

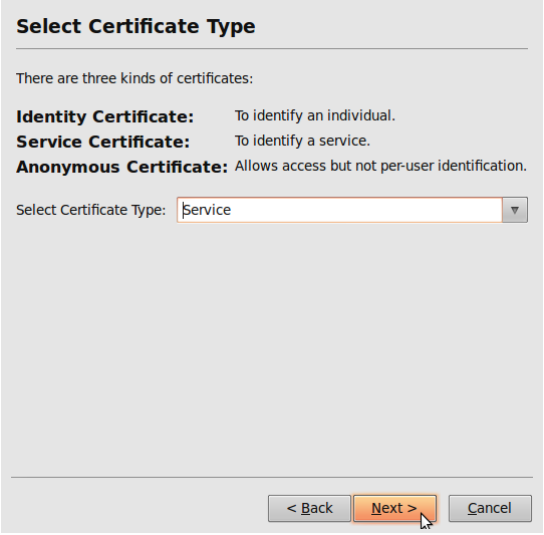


Figura 4.10: Selección del tipo de certificado a solicitar

Existen tres tipos de certificados:

- **Service**
- **Personal**
- **Anonymous**

Para nuestro servidor de pruebas, basándonos en la documentación de la página oficial de *Access Grid* [Agr10], se eligió el tipo **Service**. Este tipo de certificados es el que hay que utilizar cuando queremos utilizar un equipo como **Servidor de Salas**.

Luego, para pruebas internas, se puede solicitar un certificado **Anonymous**, es decir, un certificado anónimo. Según la documentación, se recomienda utilizar este tipo de certificados **para pruebas internas**, ya que no ofrece ningún tipo de seguridad, ni de control de acceso, por lo que el Servidor de Salas no estaría seguro.

Por último, los certificados tipo **Personal**, son certificados para los **Clientes de *Access Grid***, para controlar el Acceso.

Arranque Automático

El personal de la Universidad de Cádiz nos habían solicitado que estudiásemos cómo conseguir que el Servidor de Salas se ejecutase automáticamente al iniciar el sistema. Esta necesidad surgió a partir del objetivo de migrar el Servidor de Salas del Campus de Cádiz, al Campus de Puerto Real, puesto que estaría todo el día encendido para ofrecer el servicio las 24 horas del día. Si el equipo, por tareas de mantenimiento, o por un fallo eléctrico, u otro motivo, se reiniciara, el Servidor de Salas debería de seguir operativo.

Debido a los buenos resultados que ha tenido el Servidor de Salas de prueba, cuando se migró el Servidor de Salas a Puerto Real, se instaló también el Sistema Ubuntu. Por ello, únicamente nos centramos en investigar cómo ejecutar el sistema *Access Grid* al arrancar el sistema en Ubuntu. Ésto se consigue construyendo un **demonio de arranque**.

El aprendizaje para realizar demonios de arranque fue relativamente sencillo. Un posible esqueleto sería el siguiente:

```
1. - #!/bin/bash

2.- # Comentarios

3.- Ordenes que se ejecutaran en todos los casos

4.- start() | stop() | restart() | ... (tantas funciones como considere
    necesarias)

5.- #Llamadas
    case "$1" in
        start)
            # Codigo a ejecutar cuando llamemos al script con start:
               nombre_script start
            ;; # Dos ; para indicar fin de este bloque
        stop)
```

```
15              # Codigo a ejecutar cuando llamemos al script con start:
                    nombre_script stop
16              ;; # Dos ; para indicar fin de este bloque
17          restart)
18              # Codigo a ejecutar cuando llamemos al script con start:
                    nombre_script restart
19              ;; # Dos ; para indicar fin de este bloque
20          ...
21
22          *)  # Para otros casos que no sean los anteriores (recomendado para
                mensajes de error, etc)
23              echo $"Formas de uso: nombre_script {start|stop|restart}"
24  6.- esac
25
26  7.- exit $RETVAL
```

Donde:

1. Indicamos el intérprete que queremos utilizar.
2. Los comentarios pueden estar en cualquier lugar del script. Empiezan por '#' cada línea de comentario.
3. Se introducirán las órdenes "generales" que se ejecutarían en cualquier caso (ir a una carpeta concreta, crear ficheros temporales, etc.)
4. Es el conjunto de funciones para cada uno de los casos que queremos que se dé en nuestro script. Los más usuales son *start*, para ejecutar el script, *stop*, para detenerlo y *restart* para reiniciarlo.
5. Es el bloque donde se analiza con qué función hemos llamado el script.
6. Fin del bloque "case"
7. Se retorna el valor obtenido en el script

Además, hay que indicar una serie de datos para su correcta ejecución:

- Nombre del usuario del sistema que ejecuta el Servidor de Salas
- Directorio personal de dicho usuario (/home/usario)
- Ruta del ejecutable del Servidor de Salas, ya que, dependiendo del sistema, puede variar. Por ejemplo, en Ubuntu es `/usr/bin/VenueServer3.py` , sin embargo, en sistemas basados en RPM, como RedHat, suele ser `/usr/bin/VenueServer`
- Fichero de bloqueo, como indicador de que ya se está ejecutando el Servidor de Salas, éste se elimina al cerrar la aplicación

Todos estos datos pueden almacenarse como **variables** en un fichero de configuración.

Los ficheros de arranque desarrollados aparecen como apéndice de esta documentación, en la sección *Scripts de Arranque*.

El Gestor de Salas

El siguiente objetivo a realizar era saber gestionar el Servidor de Salas. Entre la cantidad de funciones que existe para gestionar las Salas, dentro de una Sala, **se pueden acceder a otras Salas**, conocidas como *Salidas*. Ésto permite organizar mucho mejor el Servidor de Salas.

Para ello, entre el conjunto de aplicaciones que conforman el sistema *Access Grid*, existe una herramienta llamada ***VenueManager*** (es decir *Gestor de Salas*).

En dicha aplicación, al introducir una dirección de un Servidor de Salas, podemos configurar tanto el Servidor de Salas, como gestionarlo, creando nuevas salas, editando las existentes, o borrando aquellas que no nos interese que estén.

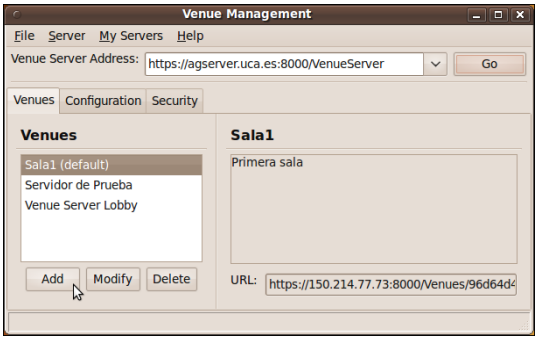


Figura 4.11: El Gestor de Salas de Access Grid

De forma resumida, el Gestor de Salas ofrece multitud de funciones tales como permitir encriptar todo flujo de datos que viaje desde o hacia el Servidor de Salas. También nos permite establecer un direccionamiento estático o dinámico para las Salas que se vayan creando. Por último, y lo que consideramos más importante, el Gestor de Salas es la herramienta a utilizar para **Controlar el Acceso** tanto del Servidor de Salas, como a una Sala concreta.

Toda esta información se encuentra en el Apéndice de esta documentación, en el Manual Completo de *Access Grid*, o en la wiki comentada anteriormente.

Control de Acceso

Como ya se comentó, uno de los principales objetivos que se estableció en las primeras reuniones fue el poder **controlar el acceso a los usuarios**, denegando dicho acceso a aquellos usuarios que no tengan permiso.

Como puede comprobar en la sección anterior, *Access Grid* dispone de una herramienta propia para controlar el acceso.

El *Gestor de Salas (VenueManager)* incluye una pestaña llamada **Security**, que es donde gestionaremos el control de acceso:

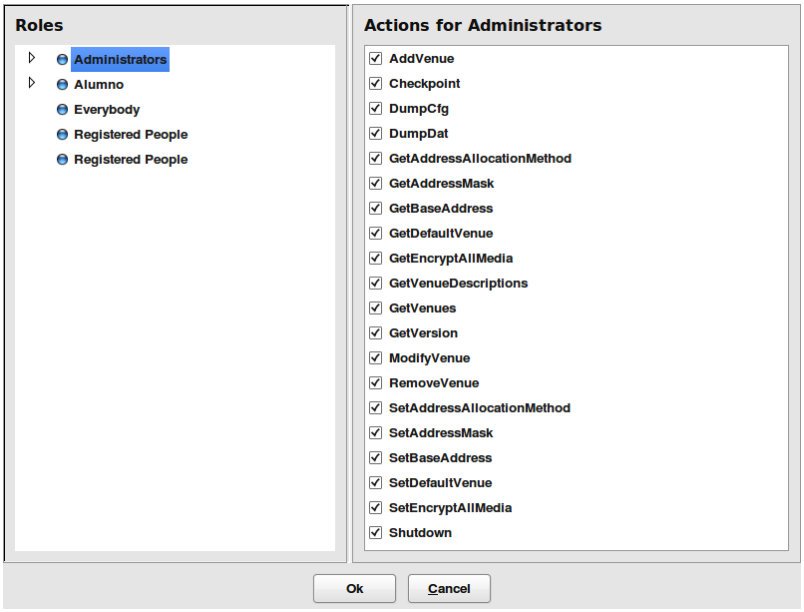


Figura 4.12: Control de Acceso en Access Grid

En dicha pestaña se pueden observar un listado de **Roles**, además de un listado de **Permisos**, llamados *Actions*. Por último, existe un listado de usuarios (**Participantes**), asignados a un Rol. El funcionamiento es muy sencillo, se pueden crear tanto nuevos Roles, como nuevos Participantes, asignándoles, luego, a un Rol. Luego, a cada Rol, se le pueden activar o desactivar permisos (*Actions*), por lo que, un Participante, dependiendo del Rol que tenga, tendrá unos u otros permisos.

Se puede controlar el acceso tanto de una Sala, como del propio Servidor de Salas. Para controlar el acceso hay que asegurarse de activar la casilla **Require user to present certificate**:

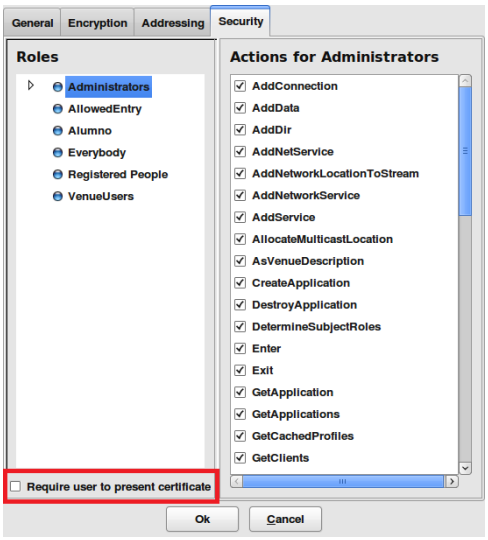


Figura 4.13: Opción que obliga al usuario presentar su certificado digital

Luego, hay que añadir al Participante que se quiera controlar su acceso, introduciendo su ***Distinguished Name***. La autoridad certificadora (AC) emite el certificado una clave pública a un DN. El DN

es la identificación única de un certificado, compuesta de varios atributos. La forma de un DN es una secuencia de pares *atributos=valor* separados por coma o por un espacio y el orden en el que aparecen los pares es importante. Estos atributos reciben el nombre de **Relative Distinguished Names (RDN)**. Los atributos existentes son:

- **DC:** domainComponent. Componente del dominio
- **CN:** commonName. Nombre común
- **OU:** organizationalUnitName. Nombre de unidad de la organización
- **O:** organizationName. Nombre de la organización
- **STREET:** streetAddress. Domicilio
- **L:** localityName. Localidad
- **ST:** stateOrProvinceName. Provincia
- **C:** countryName. País
- **UID:** userId. Identificación del usuario

Por lo que un certificado digital, puede contener el siguiente DN:

```
1 /O=Access Grid/OU=agdev-ca.mcs.anl.gov/OU=agserver.uca.es/CN=Jesus Cea
     Oliva
```

El Participante que se quiere controlar su acceso debe tener instalado un certificado de tipo Personal. Para ello, recordemos que, a la hora de solicitar un certificado al equipo de *Access Grid*, en uno de los pasos se nos preguntaba qué tipo de certificado se deseaba solicitar y, entre las opciones, se encontraba el tipo *Personal*.

El objetivo de estos certificados es que cada usuario **tenga su certificado Personal que lo identifique**. Una vez que *Access Grid* valide su solicitud, el usuario instalará su certificado en su Cliente *Access Grid* y notificará al administrador del Servidor de Salas al que desea conectarse su *Distinguished Name*, así, el administrador le dará acceso a aquellas Salas donde deba poder entrar, introduciendo su DN.

Por otro lado, para acceder a una Sala segura hay que crear un **certificado de tipo Proxy**, al crearla, *Access Grid* nos solicitará la contraseña de nuestro certificado personal. Este tipo de certificados crea una especie de ”intermediario”, ocultando así el certificado personal del usuario, protegiéndolo aún más.

Para cumplir este objetivo tuvo que resolverse muchos problemas y dudas. En un principio se pensaba que *Access Grid* no permitía controlar el acceso, por lo que el objetivo inicial era desarrollar una herramienta que se integrase con *Access Grid* y ésta se encargase de controlar el acceso.

Otra duda que teníamos era **cómo identificar un Participante**. Luego supimos que *Access Grid* gestiona el control de acceso a través de certificados digitales y éstos se identifican por el *Distinguished Name*, por lo que, introduciendo el DN del usuario, lo identificamos.

Sin embargo, existen algunos problemas que se desean solventar en un futuro no muy lejano. Por un lado, el proceso de gestionar el control de acceso nos parece muy tedioso. Todo usuario debe notificar

al administrador su DN para que éste pueda dar o no acceso a una Sala o al Servidor de Salas. No todos los usuarios deben saber cómo es su DN, además, el administrador debe de introducir ”a mano” dicho DN en la aplicación *Venue Manager*. Lo ideal sería disponer, de alguna forma, dicho DN al conectarse. Así, al ejecutar la aplicación *Venue Manager*, el administrador vería, automáticamente, un **listado de usuarios conectados** y , con arrastrarlos al Rol oportuno, se controlaría su acceso.

Es un objetivo que se ha definido para realizarlo próximamente, como mejora de código del sistema de *Access Grid*.

Gestionar nuestros propios Certificados

El proceso de solicitud y aceptación de un certificado digital por parte de *Access Grid* es un proceso manual, es decir, dicha solicitud lo recibe el personal de *Access Grid* y ésta persona acepta la solicitud, validando el certificado solicitado. Este proceso, según los propios desarrolladores de *Access Grid*, puede durar, **como mínimo, tres días**. Es decir, como muy pronto, en tres días tendríamos el certificado solicitado.

Esto supone un serio problema ya que, por un lado, dependemos de alguien externo a nosotros para que nos validen los certificados digitales de los usuarios que queremos controlar su acceso. Por otro, tres días, como mínimo, es mucho tiempo de espera. Es más, algunos certificados que hemos solicitado han llegado a **tardar más de 15 días**, por problemas meteorológicos de aquella zona.

La aplicación *Certificate Manager*, **permite importar nuevas Autoridades Certificadoras**, dentro de la pestaña ***Trusted CA Certificates***, para así, poder importar nuestros propios certificados, ajenos totalmente de *Access Grid*. Por lo que nos dedicamos a **crear nuestra propia Autoridad Certificadora** y, que, además, **emita sus propios certificados digitales**.

La Autoridad Certificadora debe estar instalado tanto en nuestro Cliente, como en el Servidor de Salas a conectar. El motivo es sencillo, en nuestro Cliente es necesario tener instalada nuestra CA creada, ya que, sin ella, *Access Grid* no es capaz de validar nuestro certificado personal y, por tanto, no podremos utilizar el Cliente. Por otro lado, el Servidor de Salas donde vayamos a entrar también necesita tener instalado nuestra CA, ya que, sin ella, el Servidor de Salas no podrá validar nuestro certificado y, por tanto, no nos dejará acceder a ninguna Sala.

Crear nuestra Autoridad Certificadora

Para crear nuestra CA y emitir nuestros propios certificados digitales, utilizaremos una herramienta que, en el mundo de Internet, es ampliamente conocida y utilizada. Se trata del conjunto de aplicaciones de ***OpenSSL*** [**?**] [2].

Lo primero que hay que hacer es crearnos la Autoridad Certificadora, la cual, recordemos, será la encargada de emitir nuestros propios certificados digitales. Para ello, ejecutamos el siguiente comando:

```
openssl req -x509 -newkey rsa:2048 -keyout ca_key.pem -days 3650 -out ca_cert.pem
```

OpenSSL es una herramienta muy versátil y nos permite generar cualquier tipo de certificado con cualquier tipo de restricción. Las indicaciones que nosotros hemos dado son:

[2] `http://www.openssl.org/`

- El certificado generado sea de tipo x509 [3]
- La clave de encriptación será de tipo RSA de 2048 bits
- El certificado y la clave se generarán en ficheros separados (ca_cert.pem y ca_key.pem respectivamente)
- La Autoridad Certificadora tendrá una validez de 10 años

Por otro lado, *Access Grid* nos solicita un fichero de tipo ***signing policy***. Este fichero es un fichero de texto normal y corriente, que contiene una serie de ”indicaciones” para que el sistema sepa cómo tratar los certificados digitales que la Autoridad Certificadora emitirá. Nosotros creamos el siguiente fichero (llamado *config_ca.signing_policy*:

```
basicConstraints = critical,CA:FALSE
extendedKeyUsage = serverAuth
```

Ahora vamos a crear un certificado digital, es decir, con nuestra propia Autoridad Certificadora. Para ello, primero generamos la clave privada para nuestro certificado digital que vamos a generar:

```
openssl genrsa -des3 -out serv-priv.pem -passout pass:aguca 2048
```

Con esta orden, estamos indicando que genere la clave privada utilizando el algoritmo *DES3* de 2048 bits, el password sea ”aguca” y se almacene en el fichero *serv−priv.pem*.

Antes de hacer un certificado , hay que hacer una petición donde se define el propietario del certificado.

```
openssl req -new -subj "/C=ES/ST=Cadiz/L=San Fernando/CN=Gestor de
        Certificados de Access Grid/OU=CITI/O=uca.es"
        -key serv-priv.pem -passin pass:aguca
        -out petic-certificado-serv.pem
```

Con esta orden estamos creando nuestro ***Distinguished Name*** (DN), introducimos el password de la Autoridad Certificadora y le indicamos el fichero de salida de la petición solicitada.

Por último, introducimos el siguiente comando:

```
openssl x509 -CA ca_cert.pem -CAkey ca_key.pem -req
    -in petic-certificado-serv.pem -days 15
    -extfile config_ca.signing_policy -sha1 -CAcreateserial
    -out servidor-cert.pem
```

Con todos estos pasos, podemos instalar nuestra Autoridad Certificadora y, además, instalar el Certificado en nuestro Servidor de Salas. Con los ficheros *ca_cert.pem* y *config_ca.signing_policy* ya se puede **importar en *Access Grid* nuestra Autoridad Certificadora**. Para ello, se ejecuta la aplicación Certificate Manager y nos dirigimos a la pestaña *Trusted CA Certificates*:

[3] http://es.wikipedia.org/wiki/X.509

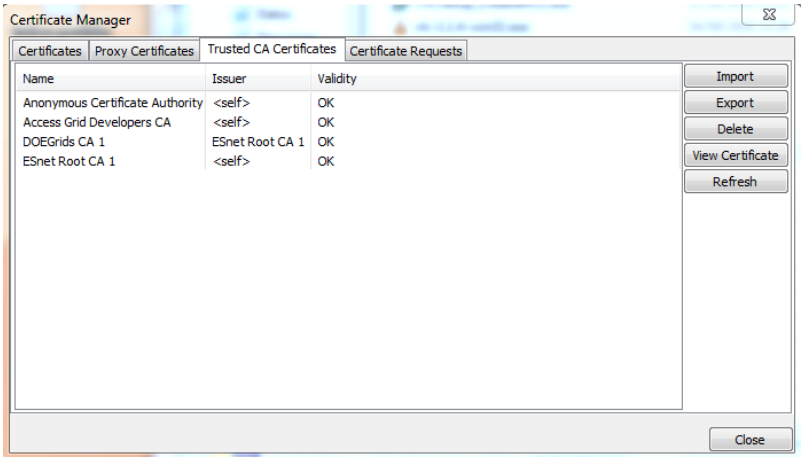


Figura 4.14: Autoridades Certificadoras de Access Grid

Pulsamos el botón **Import** para importar nuestra Autoridad Certificadora y nos aparece la siguiente ventana:

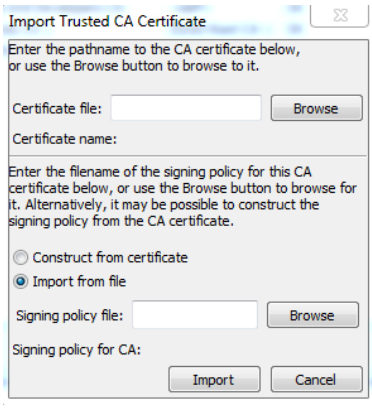


Figura 4.15: Importando nuestra Autoridad Certificadora

En ”Certificate file”, indicamos la ruta del fichero *ca_cert.pem* y en ”Signing policy file” indicamos la ruta del fichero *config_ca.signing_policy*, por último, pulsamos el botón **Import**. Con esto, **se ha instalado nuestra Autoridad Certificadora**. Estos pasos hay que hacerlos tanto en el Servidor de Salas, como en cada Cliente.

Certificado del Servidor de Salas

Para instalar el certificado digital generado en el Servidor de Salas, necesitamos los ficheros *servidor-cert.pem* (que sería el fichero de certificado) y *serv-priv.pem* (clave privada del certificado). Simplemente importamos el certificado digital en la pestaña *Certificates* y le indicamos estos ficheros en los correspondientes campos.

Emitir certificados a los Clientes

Ahora se explica los pasos que se siguió para generar los certificados cliente, en los que, en un hipotético caso, emitiríamos a los mismo para que puedan acceder a nuestro Servidor de Salas.

Primero generamos la clave privada del cliente:

```
openssl genrsa -des3 -passout pass:uca -out client-priv.pem 2048
```

La clave privada es "uca" y se genera con el algoritmo *DES3* con 2048 bits, el fichero generado es *client-priv.pem*. Ahora generamos la petición del certificado:

```
openssl req -new -key client-priv.pem -passin pass:sslfsv
    -subj "/DC=casa.com/OU=habitacion/O=uca.es/CN=cliente_aguca"
    -out petic-cert-client.pem
```

Por otro lado, creamos un fichero de tipo ***signing policy***. Nosotros creamos el siguiente fichero (llamado *config_client.signing_policy*):

```
basicConstraints = critical,CA:FALSE
extendedKeyUsage = clientAuth
```

Por último, generamos el certificado cliente con el siguiente comando:

```
openssl x509 -CA ca_cert.pem -CAkey ca_key.pem -req
    -in petic-cert-client.pem -set_serial 3 -days 15
    -extfile config_client.signing_policy -sha1
    -out client-cert.pem
```

Con estos pasos, importamos el certificado digital generado, indicando a la aplicación *Certificate Manager* los ficheros *client-cert.pem* y *client-priv.pem*.

4.3.5. Conclusiones

Saber configurar el Servidor de Salas ha sido también bastante complicado, sobre todo debido al Control de Acceso de usuarios, ya que fue un proceso bastante laborioso investigar cómo identificar a un usuario y cómo controlar su acceso.

El Control de Acceso permite **crearnos nuestros propios certificados** para no depender del personal de *Access Grid*. Como un futuro objetivo, se desea realizar una aplicación web, donde los usuarios que quieran conectarse a nuestro Servidor de Salas, deban registrarse. Así les generamos a cada uno su certificado digital y se lo enviamos, además de nuestra Autoridad Certificadora.

4.4. Puesta a punto del Sistema

Conforme se iba documentando y aprendiendo lo mejor posible el uso, configuración y optimización del sistema *Access Grid*, paralelamente se hacían multitud de pruebas en las Salas de Teledocencia, convocando, periódicamente reuniones con el resto de Campus para realizar conexiones internas y realizar multitud de pruebas. A la vez, nos daban libertad para poner en práctica todo el conocimiento adquirido para poder obtener el mejor resultado posible en las Salas de Teledocencia.

En este capítulo se detallará todo el proceso de implantación realizado, junto a la gran multitud de contratiempos y problemas que ocurrieron que, debido a ellos, no se pudo cumplir la planificación inicial.

4.4.1. Configuración de los Clientes

Todo conocimiento adquirido, comentado en el apartado anterior, donde se apreciaba una mejora en el funcionamiento del Cliente de *Access Grid*, lo poníamos en práctica y en funcionamiento en las Salas de Teledocencia.

Lo primero que se hizo fue configurar las tres Salas de Teledocencia para que se conectaran tanto al Servidor de Salas instalado en el Campus de Cádiz, como al Servidor de Salas que se instalaría más adelante en el Campus de Puerto Real, sin que tuvieran ningún tipo de problema. Además, se automatizó aún más la labor de los usuarios de los Cliente de Sala, a la hora de conectarse.

En nuestra configuración se añadió detalles sobre el códec a utilizar, que es el códec *H.261*, con un bitrate de **900 kbps** por cada cámara, ya que, en la configuración que estaba en las Salas, tenían un bitrate de *3000 kbps*, un valor demasiado alto (además que se envían más datos a la red) y, además, no se notaba ningún cambio apreciable en la calidad de vídeo.

La última mejora en la configuración añadida fue el que el usuario no tuviera que pulsar el botón **Talk** en la aplicación *RAT* para activar el envío de sonido.

Para configurar las otras dos Salas de Teledocencia, desde Puerto Real, se utiliza el visor de escritorio remoto ***VNC***.

4.4.2. Problemas de pixelación

Desde las primeras pruebas que se hacia en la Sala de Teledocencia, conectados con las otras dos Salas, se observó un problema de **pixelación de las fuentes de vídeo**. Resulta que si se conectan dos Salas, sean las que sean, la calidad de emisión y recepción tanto de audio como de vídeo es muy buena. En cambio, cuando se conecte una tercer Sala, el **sistema comienza a pixelar el vídeo**, perdiendo así la calidad de vídeo, ya que la pixelación es bastante notable, dejando una ”estela” de pixelación, un rastro que tarda mucho en desaparecer.

Se hicieron multitud de cambios en la configuración de *Access Grid*, la primera razón que se nos ocurría por entonces de por qué se pixelaban los vídeos era por un problema de ancho de banda. Así que se hizo varias pruebas disminuyendo la calidad de vídeo, sin mejorar los resultados. Por otro lado, nos pusimos en contacto con el Personal de Redes del C.I.T.I. y monitorizaron el tráfico de red de la Sala, observando que únicamente emitía unos *1.1 Mbps* y recibía unos *3 Mbps*, datos muy bajos, comparados con el ancho de banda de la Universidad de Cádiz (*100 Mbps*).

José Luis, uno de los miembros del Personal de Redes del C.I.T.I., nos comunicó que *Access Grid* asigna a los Clientes y sus servicios IPs aleatorias en multicast. Así que nos propuso que configurásemos el Servidor de Salas para que asignara un rango de IPs fijas y así limitar el rango de IPs, facilitando, además, la monitorización del tráfico de red. Aunque no solucionó los problemas de pixelación, si que sirvió para **solucionar un problema con el tráfico multicast**, que más adelante se detallará.

Además, el cliente de *Access Grid*, en su interfaz principal se observó que tenía un icono que indicaba el estado multicast, indicándonos, con un aspa roja, que no estábamos recibiendo tráfico multicast:

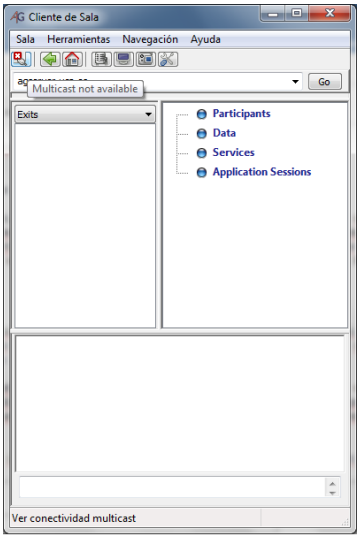


Figura 4.16: Indicador de estado de multicast

Tras un análisis detallado con José Luis, hicimos los siguientes pasos:

- Nos dirigimos a ”Venue $->$ Properties”, nos aparecerá una ventana con información de qué servicios se están ejecutando, en qué dirección y en qué puerto. Nos interesa saber la dirección y el puerto del servicio beacon:

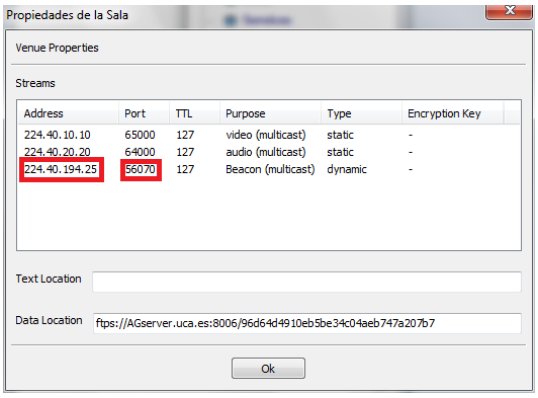


Figura 4.17: Propiedades de una Sala

- Ahora nos dirigimos a ”Tools $->$ Preferences”:

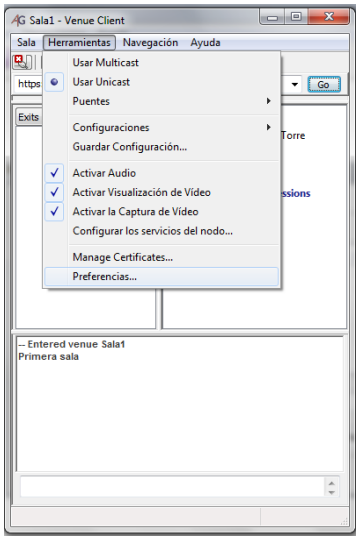


Figura 4.18: Propiedades de una Sala

- En la sección **Network** nos aseguramos de tener marcada la opción **Run integrated multicast beacon client**. Luego en *Detect multicast connectivity using the following multicast group*, escribimos en Host la dirección IP que vimos en la pestaña de Propiedades anterior, y en Port el puerto:

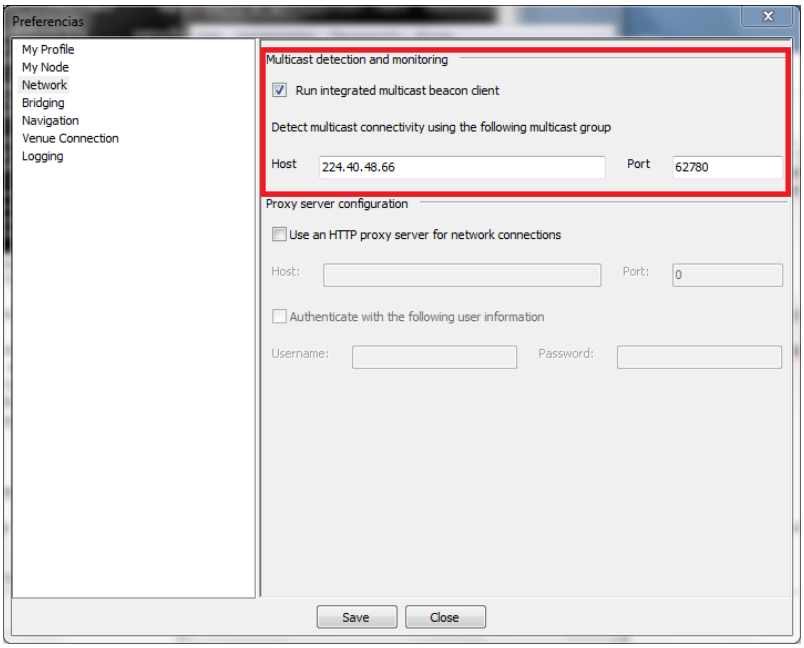


Figura 4.19: Propiedades de una Sala

- Por último, guardamos. Después de unos segundos, debería de aparecernos el icono anterior con una aspa verde, indicando que la conexión multicast es exitosa.

Se repitió este procedimiento, a través de VNC, en el resto de máquinas de Algeciras y La Bomba. Luego se ejecutó nuestra configuración en todos los equipos, asegurando que todos estábamos trabajando

igual. Y, por último, realizamos pruebas, realizando movimientos bruscos, para provocar la pixelación. Aunque al principio parecía que se había corregido, al cabo de unos minutos **volvieron las pixelaciones**.

Se observó, durante las constantes pruebas, que, por cada ventana de vídeo existente, existe un botón llamado **Info** en el que se despliega un menú y, entre las opciones, hay una que se llama **Stats** que no es más que una serie de estadísticas de paquetes de vídeo enviados, perdidos, etc, divididas en columnas (Actuales, Media, Totales). Se comparó nuestro número de paquetes perdidos totales con las otras dos aulas y el nuestro era considerablemente mayor (500 aproximadamente, contra 40 o 50). Es decir, aquella Sala que tuviera problemas de pixelación, **tendría una gran pérdida de paquetes**.

Por otro lado, documentándonos, vimos que ejecutar el cliente *Beacon* **no solucionaba ningún problema de red**, ya que *Beacon* no es más que una herramienta de testeo. Es decir, al indicarle unos valores de IP y puerto, la herramienta nos indicará si la red multicast funciona o no correctamente, pero no soluciona nada.

El motivo de la pixelación

Después de una gran cantidad de pruebas, de forma inesperada, José Luis observó que, cuando arrastrábamos una ventana de vídeo, para distribuirlas a lo largo de las 3 pantallas de la Sala, **ya se producían pixelaciones**, sin haber nadie conectado. Ejecutamos otra vez la ventana "Stats" y empezamos a mover una ventana de vídeo, se observó que, de perder pocos paquetes, al mover las ventanas, **se incrementaba considerablemente** el número de paquetes perdidos.

Así que se ejecutó el *Administrador de Tareas* y repetimos el proceso, observando que de los 4 núcleos que tiene el procesador del PC, la carga de trabajo subía hasta casi llegar al 100 % en un sólo núcleo, o si acaso dos, pero el resto de núcleos permanecían intactos. Es decir, de los cuatro núcleos, **sólo trabajaba uno o dos**.

Primera Prueba - Cambiar de Sistema Operativo

Se llegó a la conclusión de que Windows XP no estaba preparado para sacar provecho de los procesadores multinúcleos, de hecho, existen muchos artículos donde contrastan dicha conclusión, ya que Windows XP es un Sistema Operativo con muchos años en el mercado, y con el hardware que va apareciendo nuevo, no es capaz de sacarles provecho.

Así que, se optó por crear una partición nueva al disco duro del PC e instalarle Windows 7, ya que es un Sistema Operativo actualizado y preparado para el hardware que tiene instalado este PC. También se creó una tercera partición, para instalar también el Sistema Operativo Ubuntu, ya que, en el Servidor de Captura y Emisión de vídeo y audio (el equipo Linux), tiene una distribución *Fedora* instalada, y se observó que en dicho sistema la carga de trabajo se repartía equitativamente en los 4 núcleos. Sin embargo, no estábamos seguros de cómo trabajaría un sistema Unix con un **escritorio extendido de 4 pantallas**. No obstante, para comprobarlo, se creó la tercera partición para instalar ahí Ubuntu y ver, por un lado, si éramos capaces de instalar los drivers de las capturadoras (*Osprey 240e*), y, por otro, extender el monitor a 4 pantallas.

Con Windows 7, aunque el reparto de la carga de trabajo fue algo más equilibrado, **aún había problemas de pixelación**.

Con Ubuntu, se perdió tiempo en instalar los drivers de la tarjeta gráfica, una *Matrox*, y más tiempo se perdió en configurar el sistema para que extendiera a 4 pantallas el escritorio (el terminal más las 3

pantallas del aula). Sin embargo, aún habiendo conseguido todo ésto, y, aunque *Ubuntu* reparte equitativamente la carga de trabajo, tras unas pruebas con las otras dos Salas, **el vídeo se seguía pixelando**.

Segunda Prueba - Establecer Afinidad a un núcleo

Tras largas pruebas tanto en la Sala, como en nuestros puestos de trabajo, se dio con una posible solución, en los sistemas Windows, que mejoraba considerablemente el rendimiento de la aplicación *Vic*, que es quién se encarga de emitir y recibir el vídeo.

En el *Administrador de Tareas*, en la lista de procesos en ejecución, se puede seleccionar uno y establecer una opción llamada **Establecer Afinidad**. Esta opción permite asignarle a dicho proceso, qué y cuántos núcleos queremos que trabaje con él. Por defecto, todos los procesos en ejecución del sistema Windows tienen asignado todos los núcleos. Así que, de primeras, parecía ilógico disminuir el número de núcleos a un proceso. Sin embargo, la sorpresa fue que, asignando al proceso *Vic* que utilicé **únicamente un núcleo**, el rendimiento de éste mejoraba considerablemente, viendo como, en la ventana de "Stats", **no perdía ningún paquete**.

Sin embargo, este procedimiento es bastante laborioso, ya que, cada vez que se ejecute la aplicación *Vic* (al ejecutar *Access Grid*, cuando se entra en una sala, en una reconexión, etc.), había que establecer la afinidad a 1. No obstante, parece que los desarrolladores de *Access Grid* también descubrieron este procedimiento, ya que, en las opciones del **Gestor de Servicios del Nodo**, del servicio *VideoConsumer* (recordar que este servicio es para la recepción de vídeo), incluye una opción extra, únicamente disponible en los equipos Windows. Esta opción se llama **Processor Usage**, cuyo valor por defecto está en **All**, así, estableciendo este valor a 1 y guardando la configuración, nos ahorramos repetir todos estos pasos.

Convocamos otras pruebas con las otras Salas de Teledocencia, para ver si con esta opción, se pixelaban los vídeos. Por desgracia, aunque la calidad de vídeo mejoró bastante, **el vídeo seguía pixelándose** si poníamos en el escritorio más de 5 ventanas de vídeo. Es decir, de 1 a 5 ventanas de vídeo, no se producían pixelaciones, sin embargo, al añadir una ventana más, el sistema volvía a pixelar los vídeos.

Solución a la pixelación de vídeo

Se han realizado multitud de pruebas y multitud de alternativas para intentar solucionar este problema. Lo mejor que se ha conseguido es evitar las pixelaciones, con calidad de vídeo MPEG-4 y con un **máximo de 5 ventanas**.

Sin embargo, el personal de la Universidad de Cádiz, estaban empeñados en que **el primer día que se fundó la Sala, habían 9 ventanas en alta resolución con calidad MPEG-4**. A nosotros nos parecía algo imposible, principalmente porque **no caben 9 ventanas de alta resolución en 3 pantallas**, sin embargo ellos estaban convencidos de que sí.

Fue necesario una conversación, vía chat desde *Access Grid*, con uno de los desarrolladores de *Access Grid*, que, casualmente estaba conectado. Le comentamos el problema que estábamos teniendo en nuestras Salas. Su respuesta fue que emitir tantas ventanas de vídeo en calidad MPEG-4 es **una grandísima carga de CPU para nuestros equipos**, no era nada recomendable utilizar dicho códec en nuestra situación.

Nos comentó que ellos, en sus pruebas, suelen utilizar un códec para la retransmisión de vídeo en HD, pero, utilizando **únicamente una fuente de vídeo**. Por lo que nos aconsejó que utilizásemos el

códec H.261 o, en otro caso, **combinar los códecs H.261 y MPEG-4**, de forma que, aquello que se quisiera mostrar a buena calidad, fuese quien utilizase el códec MPEG-4, y el resto utilizase el códec H.261.

Así pues, se realizaron varias pruebas utilizando este códec. No ofrece la misma calidad del MPEG-4, además, la resolución máxima que permite establecer es de calidad media, a 320x240 pixeles. Sin embargo, se optimizó lo máximo posible, gracias a una opción que permite establecer la calidad de vídeo:

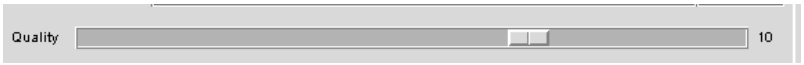


Figura 4.20: Calidad de vídeo en H.261

A más a la derecha que esté la barra, más calidad de vídeo ofrece. Así que se puso a la derecha del todo, ofreciendo la calidad máxima. El resultado fue bastante satisfactorio, ofreciendo mejor calidad que con el códec MPEG-4 utilizando la resolución media (320x240). Por lo que se empezó a configurar todos los equipos para que emitieran en H.261.

Sin embargo, en esta ocasión, a más ventanas que se ponía en el escritorio, a diferencia del códec MPEG-4, en lugar de pixelar los vídeos, lo que ocurría era que se **ralentizaban**. Es decir, la tasa de frames por segundo caía conforme se agregaban nuevas ventanas de vídeo. Reduciendo la calidad de vídeo (desplazando la barra a la izquierda), se conseguía reducir carga a la CPU y, por tanto, mejorar la tasa de frames. Así que, se consiguió **una buena configuración sin sacrificar mucho la calidad de vídeo**.

Por lo tanto, utilizar el códec H.261 en todas las cámaras con esta configuración es, por fin, ya una solución dada. Esta configuración se guardó en todos los equipos. Otra recomendación para mejorar el rendimiento (y se comprobó que funciona) es la de ”Mutear” nuestras propias cámaras. Realmente, no nos interesa vernos en nuestros propios proyectores, y mucho menos por triplicado. Así que, si se desactivan nuestras cámaras en el visualizador, son 3 fuentes de vídeo menos que tratar y, por tanto, el rendimiento mejora.

Conclusiones

Después de muchísimas pruebas, se puede decir existen soluciones antes los problemas de pixelación. Se trata de un problema de **sobrecarga de la CPU**. Por un lado, la resolución de pantalla es muy grande, 4096x768. Por otro, todo el trabajo lo realiza la CPU, ya que no hay ninguna forma de trasladar esa carga de trabajo a la tarjeta gráfica. Por último, a más **ventanas de vídeo a visualizar, más carga de CPU habría**. Por otro lado, el tamaño y la resolución del vídeo son factores también que influyen en dicha carga. Por lo que hay que ir jugando, combinando tanto el número de ventanas, como sus tamaños y resolución de emisión. Realmente, bajo MPEG-4, entre las 4 Salas de Teledocencia se pueden comunicar perfectamente, ya que no hay problemas al poner 3 ventanas de vídeo en tamaño grande, siendo una de cada Sala.

Por otro lado, no tiene sentido cargar en el visualizador nuestras propias cámaras, a lo mejor, una y de tamaño medio, a modo de comprobación de cómo nos están viendo en el exterior. Así se mejora el rendimiento a la hora de realizar una conexión con las Salas. Cargar más pantallas no tiene sentido, además que **no caben más ventanas de vídeo en las tres pantallas**, por lo que, para que quepan, se

debe reducir los tamaños de las ventanas a Medio y ésto tampoco supone un problema en el rendimiento.

Por último, durante una visita de Javier Sobrino, empleado de Auditel, para solucionar un problema de eco en la Sala de Jerez (posteriormente se detallará), nos comentó que, el día de la inauguración de las Salas de Teledocencia, **únicamente habían 2 ventanas de vídeo de tamaño grande**, a los extremos de las pantallas, y en el centro, una presentación.

Este problema ha sido el que más tiempo ha llevado solucionar y uno de los principales motivos de no haber cumplido con la planificación inicial. Sin embargo, aunque se pueda pensar de que se ha perdido mucho tiempo, el lado bueno de esta experiencia es que, gracias a ella, exprimimos al máximo *Access Grid* y aprendimos muchísimas cosas nuevas.

4.4.3. Problemas de eco. Pasos a seguir para una correcta configuración del audio

Debido a un problema de eco, que se comentará con detalle en la sección *Instalación del Aula de Jerez*, y a unos problemas técnicos de sonido en la Sala de Algeciras, Javier Sobrino, personal de Auditel y que trabajó de pleno en la instalación de las Salas de Teledocencia, vino a comprobar el estado del sonido en todas las Salas.

Javier nos dio, a nosotros los becarios, unas indicaciones de cómo establecer una correcta configuración del sonido. Unas pautas que tuviéramos que seguir, antes de comenzar algún evento o acto y asegurarnos de que el sonido, por nuestra parte, está completamente bien configurado. Todas estas indicaciones fueron documentadas y se encuentra disponible en una sección del manual completo de *Access Grid*, que se adjunta a esta documentación como apéndice, o en la wiki comentada al comienzo de este capítulo.

4.4.4. Problemas de multicast

Otro problema que tuvimos a lo largo del desarrollo de este proyecto fue debido al multicast. La Universidad de Cádiz es una de las universidades pioneras en utilizar redes multicast, sin embargo, llevan muy poco tiempo utilizándola y aún hay algunos problemas.

Por ejemplo, nuestras Salas de Teledocencia no estaban comunicándose a través de multicast ya que los cortafuegos de la Universidad de Cádiz cortaban el tráfico a nuestras redes. Gracias a José Luis y a una gran cantidad de pruebas y monitorización, se consiguió añadir los equipos de las Salas de Teledocencia para que pudieran enviar tráfico multicast.

Sin embargo, la red multicast **está limitada a la red interna** de la Universidad de Cádiz. Ésto es un problema que están resolviendo duramente el personal de Redes de la Universidad de Cádiz y, por lo visto, pronto se solucionará.

4.4.5. Uso y configuración del equipamiento de la Sala

Además de aprender a utilizar *Access Grid*, fue necesario saber utilizar todo el equipamiento disponible en las Salas de Teledocencia, para sacar el mayor partido a las Salas.

Gracias al personal de la Universidad de Cádiz, profesionales en el campo Audiovisual, se ha conseguido una formación aceptable en el uso y configuración del equipamiento audiovisual de las Salas. Desde saber manejar las cámaras domóticas de las Salas, a saber monitorizar tanto el audio como el vídeo, así como utilizar las matrices de audio y vídeo para activar aquella fuente de audio o de vídeo que

necesitásemos.

También se almacenaron unos *Pressets* tanto en las cámaras de vídeo, como en la mesa de audio, guardando la posición de las cámaras y la configuración de la mesa de sonido.

4.4.6. Migración del Servidor de Salas

Recordemos que uno de los objetivos primarios que se establecieron fue el de migrar el Servidor de Salas, alojado en el Campus de Cádiz, al Campus de Puerto Real.

El personal del Campus de Puerto Real quería **una máquina independiente** que se dedicase exclusivamente a *Access Grid* y, en general, a todo contenido que esté relacionado con las Salas de Teledocencia.

Así pues, se solicitó la adquisición de un nuevo equipo, un nuevo servidor. Análogamente, en el C.I.T.I. se estaba construyendo el **C.P.D. (*Centro de Procesamiento de Datos*)**, una Sala nueva de máquinas, actualizada con la última tecnología. La idea era alojar el nuevo equipo en esta Sala.

Hubo problemas para adquirir el equipo y tardó en llegar **varios meses**, aproximadamente a mediados de Enero se solicitó la máquina y no fue hasta Abril, aproximadamente, cuando llegó al Campus de Puerto Real.

El Sistema Operativo que se instaló fue ***Ubuntu Server 10.04***, el motivo de elegir Ubuntu fue las buenas experiencias que obtuvimos (y se sigue obteniendo) con nuestro Servidor de Salas de prueba. De hecho, mientras esperábamos a que llegase la máquina, las pruebas y reuniones que se hacían durante esos días se hacían utilizando, como Servidor de Salas, el servidor de pruebas. Otro motivo de haber elegido este sistema, es que la versión 10.04 es una versión *LAS*, ofreciendo un soporte de 5 años, a diferencia del resto de versiones que ofrecen 2 años.

El Sistema Operativo lo instalaron los administradores del C.I.T.I. y nos crearon una cuenta de usuario, con permisos de administrador. Se instaló todo el software básico, junto a *Access Grid*.

Se solicitó, una vez más, el certificado digital para *Access Grid*, además, de crear los **suscriptor de arranque** para que se ejecutara el Servidor de Salas como un demonio de arranque de Ubuntu.

Todo el proceso de puesta en marcha del Servidor de Salas, fue exitoso, si acaso, un problema bastante común que se tuvo las primeras semanas estaba relacionado con las **conexiones de red y las redes multicast**. Debido al C.P.D., todas las máquinas que se encontraban en la antigua Sala de Máquinas del C.I.T.I. fueron trasladados a la nueva sala. Ésto llevó a realizar un mantenimiento exhaustivo y, entre las tareas a realizar, estaba la asignación de direcciones IP a cada máquina, entre ellas, la nuestra. Aún así, el problema se solventó con éxito.

Por último, el Proyecto Final de Carrera de mi compañero Miguel Ángel Pérez Prado consiste en la realización de una aplicación web donde se alojarán las sesiones de *Access Grid* que se graben en las Salas de Teledocencia. No se va a entrar en detalle debido a que se trata de otro Proyecto, sin embargo, hay que comentar que todo el software necesario para dicho proyecto, y todos los vídeos que se van grabando, se encuentran almacenados en esta máquina.

4.4.7. Creación de nuestro propio Bridge

Un nuevo objetivo planteado fue que instalásemos en nuestro Servidor de Salas un ***Bridge*** multicast. *Access Grid* funciona bajo **redes multicast**, debido, principalmente a la gran cantidad de información y de paquetes de datos que viaja en dicho sistema.

Gracias a la tecnología multicast, cuando un Cliente envía datos a la red (por ejemplo, sus cámaras de vídeo), donde, por ejemplo, puede haber 6 Clientes conectados más, en lugar de tener que enviar 6 veces la misma información para cada uno de los Clientes, solamente lo envía una vez y, gracias a la tecnología multicast, los 6 Clientes reciben dicho flujo de información.

Sin embargo, **las redes multicast aún no están muy extendida**, y mucho menos a nivel doméstico. Es por ello que el sistema *Access Grid*, para aquellos Clientes que no tienen directamente conexión multicast, ofrece una alternativa, los ***Puentes Multicast*** **(Bridges)**. Estos Puentes son Servidores que sí disponen de conexión multicast. Así, se enviaría nuestro flujo de datos a dicho Puente y éste, gracias a que dispone de conexión multicast, repartiría el flujo de datos al resto de Clientes.

Access Grid permite crearnos nuestro propio Puente Multicast y que el resto de usuarios de todo el mundo lo utilicen. La Universidad de Cádiz solicitó que se creara nuestro propio Puente Multicast, así no dependíamos de servidores externos (en algunas pruebas ha habido algunos problemas con los Puentes Multicast que usualmente se utilizaban, debido a tareas de mantenimiento o simplemente se habían caído).

Para crear un Puente Multicast, basta con ejecutar un **comando** de *Access Grid* [4].

Hay que asegurarse de que, por un lado, la máquina que hará de puente dispone de conexión multicast (por este motivo se crea el puente) y, por otro, si la máquina está detrás de un cortafuegos, hay que asegurarse de tener abierto los puertos necesarios (normalmente son los puertos 20000 y usa un rango de puertos entre el 50000 hasta el 52000). El comando a ejecutar (en una terminal) es el siguiente:

```
1 Bridge.py --name=<name> --location=<location>
```

Este comando, además, ejecuta una aplicación llamada *Registry*, que lo que hace es notificar la existencia de nuestro puente, para que el resto del mundo puedan localizarlo.

Al tratarse de un comando, al reiniciar la máquina, se pararía el Puente Multicast. Por lo que se tuvo que **desarrollar otros demonios de arranque**, para ejecutar el propio Puente Multicast.

Así pues, basándonos en el esqueleto que se explicó en el apartado anterior, los ficheros de arranque desarrollados aparecen como apéndice de esta documentación, en la sección *Scripts de Arranque*.

4.4.8. Gestión de Salas y Control de Acceso

En el nuevo Servidor de Salas se han creado varias Salas, en principio sin seguir ningún tipo de estructura o jerarquía, simplemente son varias **Salas numeradas**. Además, se ha creado una Sala exclusivamente para la Gerencia y altos cargos (Vicerrectorado, etc.).

En cuanto al Control de Acceso, **no se ha puesto en marcha**, ya que es necesario convocar una reunión con el personal de la Universidad de Cádiz para determinar varios puntos importantes. Por

[4] http://www.accessgrid.org/node/125

ejemplo, qué Roles van a ser necesarios (¿Alumnos y Profesores? ¿o algún Rol más como Gerencia, Dirección, etc.?), qué permisos se desea que tengan cada uno de estos Roles, qué Salas se desean crear y cuáles de ellas se desea restringir el acceso.

Estos puntos son importantes a tratar ya que, además de la gestión interna del sistema *Access Grid*, es determinante para desarrollar uno de los futuros objetivos, la aplicación web donde los usuarios podrán registrarse, conseguir su certificado digital y, automáticamente, asignarle los permisos oportunos, según el tipo de usuario que se ha registrado.

4.4.9. Copias de seguridad

Una vez resuelto todos los problemas y habiendo configurado y optimizado las Salas de Teledocencia, se optó por realizar una copia de seguridad de todos los equipos, para recuperarlos en caso de que hubiera problemas en el funcionamiento de los equipos.

Por otro lado, hemos contactado con los Administradores de Sistemas del C.I.T.I. para establecer las políticas de copia de seguridad del Servidor de Salas migrado en el Campus de Puerto Real. Tuvimos que indicarles qué carpetas necesitábamos que se hicieran copia de seguridad. Nosotros creamos un **script de copia de seguridad de la base de datos** de la aplicación web que está desarrollando Miguel. Esta copia se realiza semanalmente, los Domingos a las 12 de la noche.

Tenemos aún pendiente de realizar una política de copia de seguridad de los equipos de las Salas de Teledocencia, ya que, hasta ahora, se realizan manualmente. Se pretende adquirir unos discos duros externos y que, semanalmente, se realicen copias de seguridad y se vuelquen en dicho disco duro externo.

4.5. Instalación del Aula de Jerez

En este capítulo se hablará sobre la actualización del equipamiento a instalar en la Sala de Teledocencia del Campus de Jerez y la implantación tanto de *Access Grid* como de *Adobe Connect*.

Si recordamos la planificación inicial que se hizo, se puede observar que dicha aula debía de estar preparada antes de final de año de 2009, para que, a la vuelta de las vacaciones de Navidad, estuviera ya en funcionamiento. El motivo principal de querer cumplir esta planificación es que un Máster que se estaba impartiendo en dicha aula, Máster de Agroalimentación, pudiera utilizar el nuevo equipamiento y el nuevo sistema *Adobe Connect*.

Sin embargo, surgieron varios problemas, que a continuación detallaremos, que impidieron poder cumplir esta planificación.

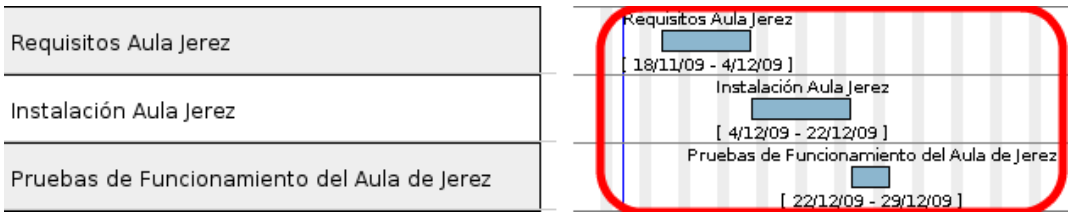


Figura 4.21: Planificación Inicial para instalar la Sala de Jerez

4.5.1. Equipo necesario

Nuestro primer objetivo fue decidir qué equipo o equipos iban a ser necesarios para el aula de Jerez. Como ya se ha comentado, la empresa Auditel actualizará el equipamiento de esta aula, por lo que se intentará reutilizar el mayor equipamiento posible, y se añadirán nuevos equipos.

La renovación de equipamiento, la agregación, luego, de nuevo equipamiento y toda su instalación en el aula de Jerez la ha llevado a cabo el personal de la empresa Auditel, a partir de los requisitos que la Universidad de Cádiz estableció.

Nuestra labor fue, a partir del equipamiento que iba a haber en el aula de Jerez, decidir qué equipo o equipos iban a ser necesarios para poder gestionar la Sala. Para ello, era necesario saber lo siguiente:

- Número de cámaras instaladas en la Sala: Para saber el número de tarjetas capturadoras necesarias.
- Número de pantallas que se conectarán al equipo: Para saber el número de tarjetas gráficas.
- Carga del Sistema: Sistemas de colaboración que se utilizarán, codificación en Windows Media, etc.

Se estableció, desde un primer momento, que únicamente se adquirirá un PC. La Sala de Jerez dispone de **dos cámaras de vídeo** instaladas (es uno de los equipamientos que se han mantenido en la renovación de equipos), por lo que serán necesarias **dos tarjetas capturadoras de vídeo**, aunque no se descarta, como futura ampliación, añadir nuevas cámaras, por lo que sería necesario una placa base con más de dos ranuras PCI Express o, en su defecto, varias ranuras PCI, para agregar futuras tarjetas capturadoras.

Por otro lado, a diferencia de las tres Salas de los otros Campus, sólo habrá una sola pantalla de videoproyección que se conectará al equipo. Sin embargo, no se descarta, como futura ampliación, añadir nuevas pantallas. **ATI** ha lanzado una nueva gama de tarjetas gráficas, la gama 5xxx, cuya característica principal que nos interesa es que integra una nueva tecnología llamada **EyeFinity** [5], la cual permite conectar **hasta seis pantallas** bajo una misma GPU, sobre todo gracias a un nuevo conector llamado **DisplayPort** [6].

Una vez decidida la tarjeta gráfica, la Universidad de Cádiz nos comunica que necesitarán solamente dos tarjetas capturadoras, de la marca **Osprey**, al haber en la Sala de Jerez dos cámaras de vídeo, pero no descartan que, en un futuro, se puedan adquirir nuevas cámaras y, por tanto, harían falta nuevas tarjetas capturadoras (como en el resto de Salas, que hay conectadas tres capturadoras, al haber tres cámaras de vídeo).

Una última preocupación era la refrigeración del equipo, al estar en constante funcionamiento, y al haber tantos componentes de alta gama funcionando en su interior. La Universidad de Cádiz decidió instalar todo el equipamiento en una caja tipo servidor, de aluminio, para refrigerar lo mejor posible. Por lo tanto, resumiendo, el equipo posee las siguientes características:

- **Placa base:** La idea es optar por una placa con el mayor número posible de puertos PCI Express, ya que las tarjetas capturadoras utilizan este puerto, además de las tarjetas gráficas actuales.
  Actualmente hay conectadas en el aula de Jerez 2 cámaras, por lo que, como mínimo, harían falta 3 puertos PCI Express, pero, si en un futuro, se quisiera añadir una cámara más (como tienen el resto

[5] `http://www.amd.com/es/products/technologies/eyefinity/Pages/eyefinity.aspx`
[6] `http://es.wikipedia.org/wiki/DisplayPort`

de Salas) haría falta una tercera capturadora con su correspondiente puerto PCI Express. La placa base suministrada es una **Intel DX50SO**, que dispone de 2 puertos PCI Express x16, 1 puerto PCI Express x4 y 2 puertos PCI Express a x1, además de incluir un puerto PCI tradicional. Tiene otras características como el socket para procesadores I7 de Intel, ranuras de memoria DDR3, sonido integrado, etc.

- **Tarjeta gráfica:** Se ha optado por una tarjeta de la nueva gama de ATI, la familia 5xxx. El motivo principal es, además de su gran potencia, el soporte de múltiples monitores en una sola GPU, hasta un máximo de 6 pantallas. El modelo solicitado es una **Asus EAH5850** que soporta hasta 4 monitores ya que tiene dos puertos DVI, 1 puerto HDMI y un puerto DisplayPort. Puede que sea necesario algún conversor para conectar los monitores y/o proyectores, pero se puede conectar 4 dispositivos en una sola GPU sin necesitar dos tarjetas gráficas.

- **Tarjetas Capturadoras:** Hace falta tantas tarjetas capturadoras como cámaras se quieran conectar en el equipo, ya que cada cámara irá conectada a una capturadora para que *Access Grid* y *Adobe Connect* capturen dicha cámara y lo retransmitan. En este caso, el aula de Jerez tiene instalada 2 cámaras de vídeo, por lo tanto, se han adquirido 2 tarjetas capturadoras. El modelo elegido es **Osprey 240e**.

- **CPU:** El microprocesador suministrado es un **I7 920** cuya potencia son 2.67Ghz.

- **Memoria RAM:** El equipo viene con **6GB DDR3 de memoria RAM** instalados en 3 módulos de 2GB cada uno.

- **Disco Duro:** El equipo viene con un disco duro de 1.5TB instalado.

- **Grabadora de DVD:** Regrabadora de DVD de doble capa.

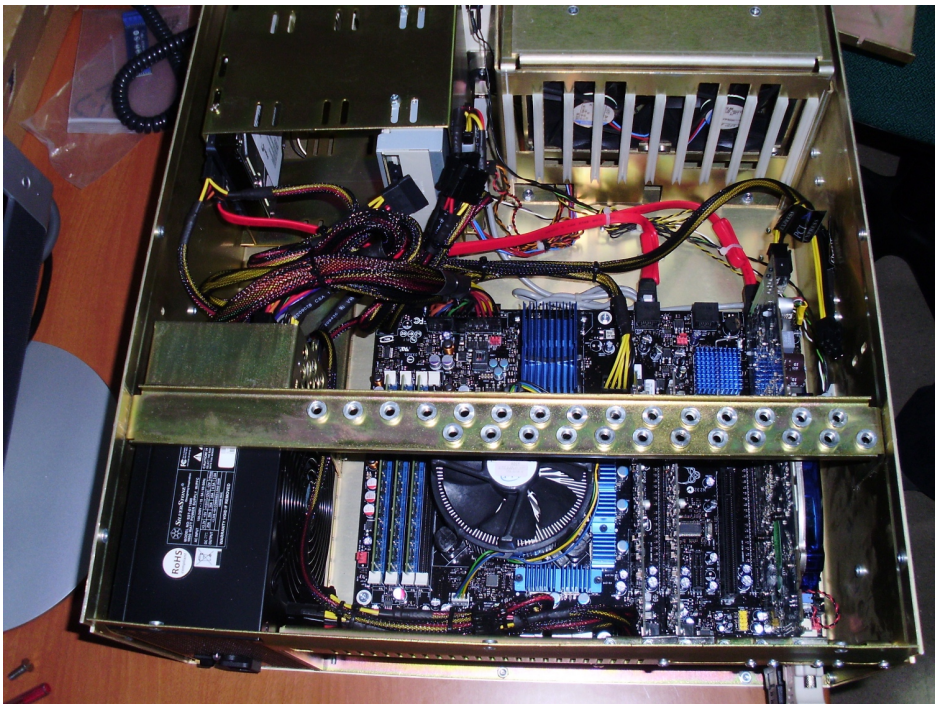


Figura 4.22: Interior del PC de la Sala de Jerez

Por último, el software que se instalará será un sistema Windows y, como sistemas de colaboración, *Access Grid* y *Adobe Connect*.

4.5.2. Problemas Surgidos

Como ya se ha comentado anteriormente, ocurrieron varios problemas que, debido a ellos, no permitieron cumplir con la planificación anterior.

Retrasos

En primer lugar, el equipo sufrió un retraso en llegar. El coste de adquisición de este equipo fue más elevado de lo planeado por la Universidad de Cádiz, ya que costó más de 4000€. El coste subió tanto principalmente por la caja elegida, tipo servidor de aluminio, para que el equipo refrigerase lo mejor posible. Después de varias discusiones, se decidió dejar dicha caja y, por tanto, adquirir el equipo por ese precio.

Por otro lado, los componentes que se eligió, según Auditel, tardaría en llegar algunos de ellos y, por tanto, no iba a ser posible tener el equipo en la fecha deseada. El PC llegó al Campus de Puerto Real a primeros de Enero de 2010, es decir, llegó el equipo mucho después de lo planificado y, por tanto, no se pudo tener todo instalado antes de fin de año.

Además, cuando llegó el PC, no llegó con todos los componentes que se solicitó; nos comunicaron que la tarjeta gráfica tardaría en llegar bastante tiempo y, mientras tanto, nos dejaron otra tarjeta de menor potencia, una ATI 3870.

Cuelgues del Sistema

Otro problema con que nos encontramos fue que el sistema se colgaba a menudo y de forma aleatoria, sin un motivo aparente. El sistema llegó al Campus de Puerto Real preinstalado, con Windows XP como sistema operativo, y el software básico para tener en funcionamiento el sistema (drivers, etc.).

No sabíamos si era un problema de hardware o de software. En un primer momento pensamos que podrían ser las memorias RAM, que hubiera alguna defectuosa, así que hicimos un test de memoria a bajo nivel, sin mostrar ningún tipo de problema. Decidimos formatear el PC para instalar nosotros por completo el sistema y, sorprendentemente, el sistema seguía congelándose aleatoriamente. Así que, casi seguro, se trata de un problema hardware.

Una vez montado el equipo e instalado todo el equipamiento de la Sala, nos dirigimos a Jerez para instalar todo el sistema. Instalamos los sistemas *Access Grid* y *Adobe Connect*, además instalamos las últimas versiones de los navegadores Internet Explorer, Firefox y Google Chrome. Por último, instalamos la aplicación VNC, para poder monitorizar el equipo remotamente, concretamente, desde nuestro puesto de trabajo en el Campus de Puerto Real.

Seguimos con problemas de congelamiento aleatorio del sistema, incluso instalamos en otra partición Ubuntu por si también se congelaba el sistema. Efectivamente, el sistema seguía congelándose sin motivo aparente, así que nos ponemos en contacto con Auditel. Éstos a su vez se pusieron en contacto con la tienda quien le suministro el PC para comunicarles el problema y dieran alguna solución. La solución que nos aportaron fue un email donde nos daban unas indicaciones sobre unos valores que era aconsejable modificar en la BIOS de la placa base. Cambiamos algunas de estas opciones, entre ellas,

disminuir la velocidad de transferencia de datos de los puertos PCI Express, por si acaso las tarjetas capturadoras eran de menor velocidad, aunque era una solución algo ilógica (se supone que si una tarjeta PCI Express trabaja a menos velocidad de la que una ranura ofrece, dicha ranura trabaja a la velocidad de la tarjeta).

Parecía que el equipo estaba estable, puesto que lo dejamos un fin de semana trabajando utilizando *Access Grid*, pero, aún así, el equipo siguió congelándose en sucesivas ocasiones.

A día de hoy, seguimos con el mismo problema y pensamos de que se trata de un problema hardware.

Problemas de eco

La Universidad de Cádiz dio luz verde para reiniciar un Máster que se quedó a la espera de la renovación del aula de Jerez. Se trataba de un Máster de Agroalimentación, impartida por el profesor Carmelo. Dicho Máster se impartía, además de en la Universidad de Cádiz, en la Universidad de Córdoba. El sistema de colaboración elegido, por solicitud propia del profesor Carmelo, fue *Adobe Connect*, ya que conocía el sistema anteriormente por uso particular y prefería usarlo. Como segunda opción, es decir, en caso de que hubiera problemas, se utilizaría, como siempre, el sistema *Polycom*.

Al comenzar estas clases, nos encontramos con un nuevo problema, relacionado con el sonido. El problema con que nos encontramos era que no conseguíamos configurar el sonido correctamente y producía eco. El Máster tuvo que suspenderse temporalmente hasta nuevo aviso.

Llegaron los técnicos de sonido de Auditel y estuvieron comprobando la conexión del cableado y la configuración de cada uno de los componentes (mesa de mezclas de audio, cancelador de eco, etc.).
Observaron que había varios errores: por un lado, había una mala conexión de cableado hacia el cancelador de eco, concretamente, la microfonía, aún estando conectada al cancelador de eco, no estaba conectada en el lugar adecuado y, por tanto, no se estaba cancelando su eco. Además, el propio cancelador de eco no estaba correctamente configurado ya que faltaban algunos patrones que configurar.

Configurar un cancelador de eco es tarea muy complicada, se necesita contratar a un especialista de sonido para que lo configure correctamente y, una vez configurado, hay que procurar no tocar nada para no desconfigurarlo. Auditel se encargó de esta labor.

Una vez configurado correctamente el cancelador de eco, realizamos varias pruebas con el resto de Salas de Teledocencia, consiguiendo un resultado más que notable.

Comenzaron de nuevo las clases del Máster de Agroalimentación y pudo finalizar con éxito.

Otros problemas

Como se puede observar, la instalación del aula de Jerez hubo muchos problemas que no se pensaba que se iba a tener. El equipo, a día de hoy, sigue congelándose aleatoriamente. La Universidad de Cádiz está intentando solventar este problema con Auditel.

Por otro lado, pensamos que existe un desaprovechamiento de la potencia del equipo por tener instalado Windows XP. El sistema Windows XP tiene varios años de vida y no aprovecha correctamente las nuevas tecnologías hardware. Por un lado, el equipo tiene un procesador I7, con 8 núcleos. Sabemos que **Windows XP no está optimizado para procesadores de más de un núcleo**, por lo que se estaría desaprovechando la potencia que ofrece el procesamiento multinúcleo. Por otro lado, el sistema posee 6

Gb de memoria RAM, pero Windows XP es un sistema de 32 bits, por lo que únicamente, podrá redireccionar hasta 3.5 Gb de memoria RAM de las 6 disponibles, es decir, que **se desaprovecha casi la mitad de memoria RAM**.

Por todo ésto, es aconsejable instalar un sistema operativo de 64 bits, ya sea Windows, en cuyo caso, recomendaríamos Windows 7, o Linux.

Resultado Final

Aún teniendo el problema de congelamiento aleatorio sin resolver, el resto de problemas se subsanaron. Así que se volvió a iniciar el Máster de Agroalimentación. En esta ocasión, gracias a los problemas que solventamos y siguiendo unas pautas que nosotros indicamos a los profesores de Jerez y de Córdoba, el Máster pudo realizarse con éxito. Una vez finalice el Máster, durante los próximos meses, se pretende solucionar el problema del congelamiento aleatorio del PC.

A continuación se muestran algunas imágenes de cómo quedó el aula de Jerez después de la renovación:

Figura 4.23: Cabina de Control I de la Sala de Jerez

Figura 4.24: Cabina de Control II de la Sala de Jerez

Figura 4.25: Armario I de la Sala de Jerez

Figura 4.26: Armario II PC de la Sala de Jerez

Figura 4.27: Sala de Jerez I

Figura 4.28: Sala de Jerez II

4.6. Certificación de las Salas de Teledocencia

Una vez estabilizada las Salas de Teledocencia y optimizadas, el personal de la Universidad de Cádiz querían **certificar las Salas de Teledocencia con *Access Grid***.

Esta certificación consiste en evaluar las Salas de Teledocencia y determinar, a partir de una serie de pruebas, si las Salas de Teledocencia están capacitadas para poder realizar sesiones de *Access Grid*. En caso de que la evaluación sea positiva, las Salas formarán parte de un listado de la comunidad de *Access Grid*, como Salas certificadas.

Esta certificación lo realiza una persona cualificada, **un certificador oficial de *Access Grid***. Dicha persona se conectaría a una sesión de *Access Grid* y dará una serie de instrucciones e indicaciones a seguir para comprobar el correcto funcionamiento de *Access Grid* y evaluar el estado de la Sala. En España, el certificador oficial es Vicente Goyanes, de la Universidad de Vigo.

Antes de ponerse en contacto con Vicente Goyanes, hay que asegurarse de que se ha comprobado todo y que todo está a punto. El proceso de evaluación consiste en comprobar que la sala cumple con una serie de requisitos que se encuadran en cuatro categorías:

1. **Audio:** Conjunto de pruebas que verifican una buena configuración de audio.
2. **Video:** Conjunto de pruebas que verifican una buena configuración de vídeo, además de las cámaras de vídeo.
3. **Networking (Redes):** Conjunto de pruebas que verifican una buena configuración de la red.
4. **Supporting Tools (Herramientas de soporte):** Únicamente comprueban que podemos acceder a los Participantes a través de escritorio remoto.

Aunque la Sala esté preparada para usar el códec H.264 (el cual no es obligatorio), el certificador solicita que se conecte a través de H.261 puesto que es el que se está homologando para utilizar en las salas.

4.6.1. Resultados de la Certificación

Una vez realizada todas las pruebas propuestas por el certificador oficial de *Access Grid*, el resultado obtenido fue **muy satisfactorio**. Como pegas, nos indicaron, por un lado, que la posición de las cámaras no permite simular esa comunicación directa que recomendaban, ya que están a una altura, que por mucho que se intente enfocar no lo permite. Por otro lado, como se ha comentado, la red multicast, de momento, está limitada al exterior, por lo que no fue posible establecer una conexión a través de multicast. Sin embargo, nos ha comunicado que cuando solventemos este problema, nos pongamos de nuevo en contacto con él para evaluar de nuevo esta característica.

Positivamente, el certificador oficial quedó muy sorprendido por **la gran calidad de vídeo conseguida a través del códec H.261**, reconociendo que nunca antes había visto una calidad de imagen como la nuestra, en ninguna otra Sala. Por otro lado, también nos felicitó por la gran calidad de audio conseguida, al **no haber ningún tipo de eco, ni ruido** en las Salas, permitiendo ello, un sonido muy limpio.

4.7. Presupuesto mínimo de *Access Grid*

Un último objetivo que se estableció en la primera reunión que hubo al comienzo de este proyecto fue la **elaboración de un Presupuesto mínimo** para que un aula o un despacho pudiera tener instalado un sistema *Access Grid* con una calidad de audio y vídeo aceptables.

Realmente, **cualquier usuario puede utilizar *Access Grid***, basta con disponer de una webcam y un micrófono. Hay que asegurarse de no emitir eco, y esto, la forma más barata que existe es la de **utilizar unos auriculares**. Así el sonido que sale del ordenador no vuelve a entrar por el micrófono y, por tanto, evitamos el eco.

Sin embargo, esta solución no es aceptable para un aula ni para un despacho, sobre todo si hay otras personas presentes en la reunión de *Access Grid*. Así que se investigó sobre qué equipamiento iba a ser necesario, como mínimo, para estos casos.

En cuanto a la webcam, cualquier de los modelos disponibles en el mercado es suficiente. Sin embargo, hay que asegurarse que la webcam elegida **es compatible con *Access Grid***, como se ha comentado anteriormente, en la página oficial de *Access Grid* existe un listado de todo el hardware que se ha probado con este sistema y los resultados obtenidos, entre el hardware disponible en el listado, se encuentran las webcams.

La webcam elegida por el personal de la Universidad de Cádiz es una *Logitech Quickcam Pro 9000* [7]. Esta webcam es de gama alta y ofrece una gran calidad de vídeo, ofreciendo la posibilidad de **capturar en Alta Definición** a una resolución de 1280x720. Su precio es de unos 90€aproximadamente [8].

[7] `http://www.logitech.com/en-us/webcam-communications/webcams/devices/6333`
[8] Imagen cogida de `http://bestpriceson.net/logitech-quickcam-pro-9000/`

Figura 4.29: Webcam Logitech Quickcam Pro 9000

En cuanto al sonido, es el elemento más importante y más crítico, ya que sin sonido no tiene sentido establecer una comunicación por videoconferencia, por muy bien que se vean las imágenes. Es necesario adquirir un sistema de microfonía que sea capaz de capturar y eliminar el eco, ya que, sin esta cualidad, se generará eco y habría problemas con el audio. El sistema de microfonía elegido por el personal de la Universidad de Cádiz es un *YAMAHA PJP-25UR* [9]:

Figura 4.30: Sistema de microfonía YAMAHA PJP-25UR

Este sistema de microfonía dispone de un **cancelador integrado**, no de la misma calidad ni la misma eficacia que tiene el cancelador de eco de una Sala de Teledocencia, pero permite que haya una comunicación audiovisual fluida y sin eco. Como se puede comprobar, está compuesto por tres módulos de micrófonos, flexibles, permitiendo ajustarlo según cómo estén situados los miembros de la Sala donde se encuentre este sistema de microfonía. Los resultados obtenidos en los despachos instalados han sido muy satisfactorios y se han realizado varias pruebas con las Salas de Teledocencia, emitiendo a una gran calidad de sonido (aunque a un volumen algo más bajo que el de las Salas) y sin emitir ningún tipo de eco. Su precio, sin embargo, es algo elevado, ya que cuesta aproximadamente unos 650€.

En cuanto al equipo, en general, **cualquier PC es adecuado para *Access Grid***, aunque, como se comentó en los problemas de pixelación, es recomendable disponer de un PC potente para que sea capaz de soportar varias ventanas de vídeo. En un principio se utilizó un ***Pundit***, es un PC de dimensiones muy reducidas con buenas características de hardware [10].

[9] Imagen cogida de `http://www.flickr.com/photos/nextspace/2631879722/`
[10] Imagen cogida de `http://parker1.co.uk/mythtv.php`

Figura 4.31: Pundit

Sus dimensiones reducidas, su emisión baja de ruido, su bajo precio (aproximadamente unos 450€), además de disponer unas características hardware muy aceptables, hacen que sea el equipo idóneo para estas Salas.

Sin embargo, el personal de la Universidad de Cádiz no estaban muy convencidos de esta solución, ya que lo veían algo ”engorroso” al tener tantos cables. Querían una solución más ”integrada”, que todo estuviera en un solo aparato y, así, quitar el mayor número de cables posibles.

La solución que encontraron consiste en una **pantalla LCD de 40” con un PC integrado en la parte de atrás**, concretamente, el modelo es un *Samsung 400UXn-UD2* [11].

Figura 4.32: Pantalla LCD Samsung con PC integrado

Se trata de una pantalla LCD en el que, en la parte de atrás, integra un PC equipado con las siguientes características:

[11] `http://www.samsung.com/es/consumer/pc-peripherals-printer/lfd-display/slim-video-wall-solutions/LH40MRTLBN/EN/index.idx?pagetype=prd_detail`

- CPU de doble núcleo a 2 Ghz
- 1 Gb de memoria RAM DDR
- 40 Gb de disco duro
- Tarjeta gráfica integrada de 256 Mb compartidos
- Tarjeta de sonido integrada
- 6 puertos USB
- Sistema Operativo Windows XP Media Center

Con sólo esta pantalla, además de la webcam y el sistema de microfonía, se puede instalar un sistema de colaboración de *Access Grid* en las Aulas y en los despachos donde lo necesiten. Sin embargo, esta pantalla tiene varios inconvenientes:

Por un lado, consideramos el **precio bastante elevado** para las características que ofrece, aproximadamente **2500 €**. Por otro lado, las características del PC nos parecen **algo limitadas**. Sobre todo la memoria RAM, ya que la tarjeta gráfica, al ser compartida, consume parte de dicha memoria, dejando al sistema libre 768 Mb. Pensamos que, adquiriendo por separado todos los elementos, es decir, una pantalla LCD de 40" y, por otra parte un buen PC con dimensiones reducidas (ya sea un Pundit o un Barebone), estamos convencidos de que el precio total es mucho menor que el precio de estas pantallas.

No obstante, de momento, el personal de la Universidad de Cádiz, ve mejor esta solución y es la que van a aplicar en despachos y aulas. Así que el precio mínimo es el siguiente:

- **Webcam Logitech Quickcam Pro 9000:** 90€
- **Sistema de microfonía YAMAHA PJP-25UR:** 650€
- **Pantalla LCD Samsung 400UXn-UD2:** 2500€
- **TOTAL: 3240€**

A modo de curiosidad, si optamos por la solución anterior, eligiendo como equipo un Pundit, y una pantalla LCD de 40 " cualquiera, aunque pondremos un precio alto (un LCD de muy buenas características), el presupuesto mínimo sería el siguiente:

- **Webcam Logitech Quickcam Pro 9000:** 90€
- **Sistema de microfonía YAMAHA PJP-25UR:** 650€
- **Pundit de altas características:** 500€
- **Pantalla LCD Samsung 400UXn-UD2:** 1000€
- **TOTAL: 2240**

Como vemos, esta solución es mucho más barata que la propuesta anteriormente.

Capítulo 5

Fase de Desarrollo

5.1. Introducción

En este capítulo se hablará sobre el desarrollo para *Access Grid*. Como ya se sabe, *Access Grid* es de código abierto y disponemos de su código fuente para su modificación y/o ampliación. Los desarrolladores de *Access Grid* se han esforzado al máximo y han realizado un gran trabajo con el código fuente del sistema, ofreciendo un código muy legible y bien estructurado, pero sobretodo, se han esforzado, porque han desarrollado una API (a partir de ahora conocido como ***Access Grid Toolkit***) para facilitarles a ellos el trabajo y, además, para facilitarnos el trabajo a nosotros, como desarrolladores, consiguiendo realizar aplicaciones fácilmente, olvidándonos de algunas premisas necesarias para el sistema.

Por ejemplo, el sistema realiza comunicaciones a través de SSL. Si los desarrolladores no hubieran desarrollado una API para facilitar la labor de desarrollo de aplicaciones para *Access Grid*, cualquier desarrollador tendría que implementar desde cero todo el proceso de comunicación a través de SSL. Sin embargo, gracias a la API implementada, utilizando las funciones que nos aportan los desarrolladores, podemos establecer comunicación a través de SSL, olvidándonos incluso del propio SSL.

Así pues, en este capítulo se explicará cómo desarrollar para el sistema *Access Grid*, desde corrigiendo su código fuente, a añadir nuevas características o desarrollando lo que se denomina Aplicaciones Compartidas.

En primer lugar, se explicará cómo configurar nuestro equipo para poder utilizar la API del *Toolkit de Access Grid*, explicando los requisitos necesarios y el procedimiento a seguir para tener a punto nuestro sistema. Luego se dará una serie de nociones sobre cómo programar directamente en *Access Grid*, utilizando sus clases, funciones y tipos de datos. Después se comentará los errores que se han ido encontrando conforme se iba utilizando *Access Grid* y cómo corregirlos. Por último, se hablará sobre la API para desarrollar Aplicaciones Compartidas, dando unas nociones de dicha API para una mejor comprensión de su código y se hablará sobre las Aplicaciones Compartidas que se han desarrollado para la Universidad de Cádiz.

5.2. Cómo crearnos nuestro marco de trabajo

Tanto para la corrección de código, como para el desarrollo de nuevas aplicaciones para *Access Grid*, es necesario realizar una serie de pasos para tener listo nuestro sistema. Por un lado, es necesario descargarnos la última versión del código fuente del sistema, pero, además, hay que establecer unas variables de entorno para que nuestro sistema pueda utilizar las librerías proporcionadas por *Access Grid*.

Access Grid está principalmente escrito en *Python*, por lo que es primordial que el desarrollador esté familiarizado con el lenguaje *Python*.

5.2.1. Comunicación entre desarrolladores

La comunicación entre desarrolladores es muy importante, en muchas ocasiones nos pueden ayudar cuando queremos modificar el código fuente de *Access Grid* o desarrollar nuevas aplicaciones para él. La página oficial de *Access Grid* han creado una comunidad de comunicación, permitiendo a las personas involucrarse en un proyecto determinado. Gracias a ésto, se conduce a mejoras en la concepción, diseño o ejecución en el desarrollo de *Access Grid*, además puede encontrar colaboradores que le ayuden en la codificación de sus aplicaciones. Por ello, es aconsejable que se registre en los siguientes mailing-lists:

- http://www.accessgrid.org/mailinglists: Esta lista es la principal para los desarrolladores, es la lista más activa.
- http://bugzilla.mcs.anl.gov/accessgrid: Esta lista sirve para comunicar errores encontrados en *Access Grid*.
- http://www.accessgrid.org/retreat/current: Son eventos donde los desarrolladores comparten sus ideas y sus nuevos proyectos.

5.2.2. Estándares en la codificación

Si se desarrolla aplicaciones para *Access Grid*, o se modifica su código fuente, es aconsejable seguir unos estándares y estilos en la implementación de código para que todo el mundo pueda leer sin problemas dicho código fuente.

Principalmente se aconseja seguir el estilo de programación de *Python* [1]. Sin embargo, se aconsejan otras pautas:

- **No subir código fuente con tabulaciones:** Es aconsejable no subir ficheros con tabulaciones, ya que, cada usuario puede tener configurado su editor preferido, con un nivel de tabulación diferente (2 espacios, 4 espacios, 8 espacios, etc.) y puede que un código, aparentemente legible para un usuario, sea totalmente ilegible para otro usuario. Muchos editores dispone de una opción que, si se activa, al pulsar la tecla de tabulación, lo sustituye por espacios, es decir, si por ejemplo, tenemos configurado que al pulsar la tecla de tabulación avance 2 espacios, si tenemos la opción activada, en lugar de introducir una tabulación de 2 espacios, introducirá dos espacios. Así, esté como esté configurado el editor de textos, en cualquier PC, aparecerá el código con 2 espacios de separación.
- **Subir los ficheros en formato UNIX, no en Windows/DOS:** Por varios motivos, pero la principal de ellas es la coherencia.

5.2.3. Instalando el Entorno de Desarrollo

A continuación detallaremos los pasos a seguir para instalar en su equipo el entorno de desarrollo para desarrollar para *Access Grid*.

Si no tiene instalada una versión oficial de *Access Grid*, es necesario seguir estos pasos para instalar una serie de paquetes dependientes.

[1] `http://www.python.org/dev/peps/pep-0008/`

Instalación de Python y wxPython

Como se ha comentado, *Access Grid* principalmente está desarrollado bajo Python, así, es necesario tener instalado *Python*, concretamente la versión 2.4, ya que es la versión en la que se ha desarrollado este sistema.

En la mayoría de sistemas Linux ya viene Python instalado por defecto, así que no es necesario hacer ésto si va a desarrollar con una máquina Linux. En máquinas Windows podemos descargarnos el instalador desde su página oficial, concretamente en `http://www.python.org/ftp/python/2.4.4/python-2.4.4.msi` para máquinas Windows de 32 bits y `http://www.python.org/ftp/python/2.4.4/python-2.4.4.ia64.msi` para máquinas Windows de 64 bits.

Una vez instalado, procederemos a instalar *wxPython*. *wxPython* es el paquete *wxWidgets* transportado a *Python* para desarrollar aplicaciones de ventanas con esta interfaz en *Python*. Las ventanas de *Access Grid* están desarrolladas con *wxPython*, por eso es necesario instalar este paquete.

Para instalar wxPython en Linux basta con escribir la siguiente sentencia:

```
sudo apt-get install python-wxgtk2.8 python-wxversion
```

Para instalar wxPython, nos descargamos el instalador de la siguiente dirección `http://sourceforge.net/projects/wxpython/files/wxPython/2.8.10.1/wxPython2.8-win32-ansi-2.8.10.1-py24.exe/download`.

Por último, para máquinas Windows, es necesario instalar el paquete **Python Windows Extensions**, que se puede descargar de la siguiente dirección `http://sourceforge.net/projects/pywin32/files/pywin32/Build%20214/pywin32-214.win32-py2.4.exe/download`.

Obtención del código fuente

Una vez instalado Python y sus dependencias, lo primero que hay que hacer es obtener el código fuente del sistema *Access Grid*. Si lo desea, puede visualizar este código visitando la página `https://trac.ci.uchicago.edu/accessgrid/browser` [2].

Para obtener una copia del código fuente, es necesario disponer de la herramienta ***Subversion***. Una vez tengamos instalado *Subversion*, nos descargamos una copia del código fuente con el siguiente comando:

```
svn co https://svn.ci.uchicago.edu/svn/accessgrid/trunk AccessGrid
```

Ésto llevará algún tiempo, dependiendo de la velocidad de su línea contratada de Internet (y de la respuesta del servidor, ya que, a veces, suele estar saturado). Una vez haya finalizado, ya dispondrá del código fuente del sistema *Access Grid*.

[2] Puede que su navegador le dé un aviso de que el certificado de seguridad de la página esté caducado y que puede que no sea seguro entrar en dicha página, no se preocupe, la página es completamente segura.

Instalación del paquete ZSI

A continuación, debemos descargarnos e instalarnos el paquete llamado ***ZSI*** *(The Zolera Soap Infrastructure)*, ZSI es un paquete de Python que proporciona una implementación de mensajería a través de SOAP, dicho paquete lo utiliza el sistema *Access Grid*, de ahí que sea necesario instalar este paquete. Podemos descargarnos dicho paquete de su sitio oficial, `http://pywebsvcs.sourceforge.net/`, pero, concretamente, en `http://sourceforge.net/projects/pywebsvcs/files/ZSI/ZSI-2.0/` disponemos de la última versión del paquete.

Una vez descargado, lo descomprimimos en cualquier lugar y, desde una terminal, nos dirigimos a la carpeta descomprimida, por último, instalamos el paquete *ZSI*:

```
cd /ruta_carpeta_descomprimida
python setup.py install
```

Estableciendo las variables de entorno

Para finalizar la instalación del entorno de desarrollo, debemos indicar a nuestro sistema, dónde están alojadas las librerías de *Access Grid* para poder utilizarlas en nuestras aplicaciones. Para ello hacemos lo siguiente:

- **En Windows:** set PYTHONPATH=`C:\ruta_codigo_fuente_AccessGrid`
- **En Linux:** export PYTHONPATH=/home/usuario/ruta_codigo_fuente_AccessGrid. [3]
- **En OSX:** . /Applications/AccessGridToolkit3.app/Contents/Resources/setupenv.sh
  export PYTHONPATH=/Users/usuario/ruta_codigo_fuente_AccessGrid

Por último, y para finalizar, ejecutamos el script que viene acompañado del código fuente de *Access Grid* para instalar el resto de variables de entorno:

```
cd /ruta_codigo_fuente_AccessGrid/boot
python bootstrap.py
```

Verificación de que se ha instalado correctamente el Entorno de Desarrollo

Existen varias formas para verificar que se ha instalado correctamente todo este proceso, como por ejemplo, escribir un pequeño programa de prueba y comprobar que, efectivamente, *Python* lo ejecuta sin problemas. Sin embargo, lo más fácil, fiable y rápido para saber que se ha realizado todos estos pasos correctamente es ejecutar alguna aplicación ya descargada del propio código fuente del sistema *Access Grid*. Para ello, desde una terminal, nos dirigimos a la carpeta donde se encuentre el código fuente y ejecutamos alguna de estas aplicaciones:

```
cd /ruta_codigo_fuente_AccessGrid
python bin/VenueClient.py
python bin/VenueServer.py
```

Si se ejecutan estas aplicaciones, el entorno de desarrollo está instalado correctamente.

[3] Para que quede permanentemente configurada la variable de entorno, se recomienda añadirla al fichero de variables de entorno almacenado en /etc/enviroment

Sitio web para desarrolladores

Para más información acerca de cómo instalar el entorno de desarrollo y otro tipo de documentación, visite la página oficial de *Access Grid* para desarrolladores, `http://www.accessgrid.org/developer`.

Documentación de Referencia

La documentación generada a partir del código fuente de *Access Grid* se encuentra en [Vis05].

5.3. Introducción al Desarrollo en Access Grid

El conjunto de librerías que proporciona la API del sistema *Access Grid* es enorme, existiendo una gran cantidad de clases, tipos de datos y funciones para que el desarrollador pueda crear aplicaciones lo más fácil que sea posible. Sin embargo, la documentación generada a partir del código fuente del sistema *Access Grid* [4], está algo desorganizada y puede dificultar a la hora de que el desarrollador intente formarse en el desarrollo para *Access Grid*.

Es por ello que, en este capítulo, no se pretende explicar toda la API que *Access Grid* ofrece (haría falta un gran número de volúmenes para documentar todas sus funcionalidades), sino que se intentará, a partir de la documentación oficial sobre el desarrollo en *Access Grid*, además de otros documentos ([Aws04]), dar unas nociones para que el desarrollador se introduzca en el mundo del desarrollo de aplicaciones para *Access Grid* y, una vez introducido, por cuenta propia, se forme más profundamente en el desarrollo para este sistema, hasta que el propio desarrollador considere adecuado.

5.3.1. Prerrequisitos

Además de los pasos indicados en el capítulo 5.2 para introducirse en el desarrollo en *Access Grid*, en la versión 3.x [5], a la hora de desarrollar aplicaciones, puede ser útil utilizar un módulo incluido en la API *Access Grid*, para ayudar a localizar el conjunto de librerías Python de *Access Grid*.

El siguiente código ejemplo selecciona la versión 3 de *Access Grid* y, gracias a ello, hace que Python apunte a la ruta donde esté instalada esta versión, antes de seguir procesando las siguientes instrucciones.

```
1          import agversion
2          agversion.select(3)
```

Este mecanismo es usado por los programas incluidos en los instaladores de *Access Grid* (como por ejemplo, *VenueClient*, *VenueManagement* o *VenueServer*).

5.3.2. Inicialiación de una Aplicación Access Grid

Las aplicaciones que usen alguna de las librerías o, en general, cualquier elemento definido dentro del *Toolkit de Access Grid*, deben crear lo que se denomina una **aplicación Access Grid** e inicializarla. Ésto será necesario en aquellos casos en los que el desarrollador, por ejemplo, quiera implementar en sus aplicaciones un procesamiento de argumentos, loggin de lo que sucede en el transcurso de la vida de

[4] `http://www.accessgrid.org/files/developer/api/3.1/index.html`

[5] Actualmente, la última versión de *Access Grid* es la 3.2 beta, aunque sea una versión beta, los desarrolladores recomiendan instalar esta versión ya que corrige una serie de fallos importantes en el sistema

una aplicación, el uso de algunas de las clases de configuración, etc.

Existen dos clases para crear aplicaciones *Access Grid*, éstas son **`CmdlineApplication`** y **`WXGUIApplication`**. Por ejemplo, si queremos inicializar una aplicación `CmdlineApplication`, sería, por ejemplo, el código siguiente:

```
1          from AccessGrid.Toolkit import CmdlineApplication
2          app = CmdlineApplication.instance()
3          app.Initialize('MyApp')
```

5.3.3. Argumentos Predefinidos

La API del *Toolkit de Access Grid*, incorpora una serie de argumentos predefinidos, intentando agrupar los argumentos más usuales que se suelen utilizar a la hora de desarrollar una aplicación. Toda aplicación que se cree bajo el *Toolkit de Access Grid* aceptará estos argumentos estándares. Los argumentos son:

- **−d, −−debug:** Activa la depuración del programa en ejecución
- **−l, −−logfile:** Especifica un fichero log donde almacenar toda el logging de la aplicación en ejecución
- **−−numlogfiles:** Especifica el número de ficheros log sin borrar
- **−−logfilesize:** Especifica el tamaño máximo que el fichero log puede tener
- **−c, −−configure:** Especifica un fichero de configuración para la aplicación en ejecución
- **−−version:** Imprime qué versión del *Toolkit de Access Grid* se está ejecutando

5.3.4. Loggin

La mayoría de aplicaciones de *Access Grid* supervisan su ejecución almacenando todo un historial en ficheros log, cuyos nombres corresponden al nombre de la aplicación ejecutada. Por ejemplo, la aplicación anterior escrita, generaría un fichero llamado MyApp.log.

La ruta donde se almacenan estos ficheros logs es diferente, dependiendo del sistema operativo. En caso de dudas o problemas a la hora de ejecutar o desarrollar aplicaciones para *Access Grid*, es aconsejable visualizar estos ficheros log en busca de posibles errores o información que pueda facilitar su solución. La ruta, según el Sistema Operativo, es:

- **Linux, OSX:** `~/.AccessGrid3/Logs`
- **Windows:** `C:\Documents and Settings\USUARIO\Application Data\AccessGrid3\Logs`

5.3.5. Clases de Configuración

Existen tres clases de configuración predefinidas en la API del *Toolkit de Access Grid*: `SystemConfig` [6], `AGTkConfig` [7], `UserConfig` [8]. Estas clases agrupan información sobre el sistema, la configuración del *Toolkit de Access Grid* y sobre la configuración del usuario, respectivamente.

Por ejemplo, si utilizamos la clase `SystemConfig`, uno de los métodos más usuales a utilizar es `GetUserName()` que proporciona información acerca del usuario que está ejecutando la aplicación. Si utilizamos la clase `AGTkConfig`, uno de los métodos más usuales a utilizar es `GetInstallDir()` que devuelve la ruta donde el software está instalado. Por último, si utilizamos la clase `UserConfig`, uno de los métodos más usuales a utilizar es `GetProfile()` que devuelve la ruta donde se encuentra el perfil del usuario.

5.3.6. Tipos Definidos

Como se ha comentado anteriormente, uno de los elementos que incorpora la API del *Toolkit de Access Grid* son tipos predefinidos. Algunos de estos tipos son:

- **DataDescription:** Descripción de un fichero de datos de una Sala.
- **StreamDescription:** Descripción de una dirección unicast o multicast.
- **VenueDescription:** Descripción de una Sala de *Access Grid*.
- **VenueState:** Descripción de los contenidos de una Sala de *Access Grid*.

Si desea visualizar todos los tipos predefinidos en la API del *Toolkit de Access Grid*, puede encontrarlos definidos en el fichero *Descriptions.py*.

5.3.7. Ejemplos Sencillos

A continuación, para facilitar un poco más la comprensión de la API del *Toolkit de Access Grid*, se mostrarán una serie de ejemplos sencillos donde se usan las clases y los métodos más usuales de *Access Grid*.

Para tener una primera idea sobre la arquitectura de la API de *Access Grid*, se muestra la siguiente figura:

[6] `http://www.accessgrid.org/files/developer/api/3.1/AccessGrid.Config.SystemConfig-class.html`

[7] `http://www.accessgrid.org/files/developer/api/3.1/AccessGrid.Config.AGTkConfig-class.html`

[8] `http://www.accessgrid.org/files/developer/api/3.1/AccessGrid.Config.UserConfig-class.html`

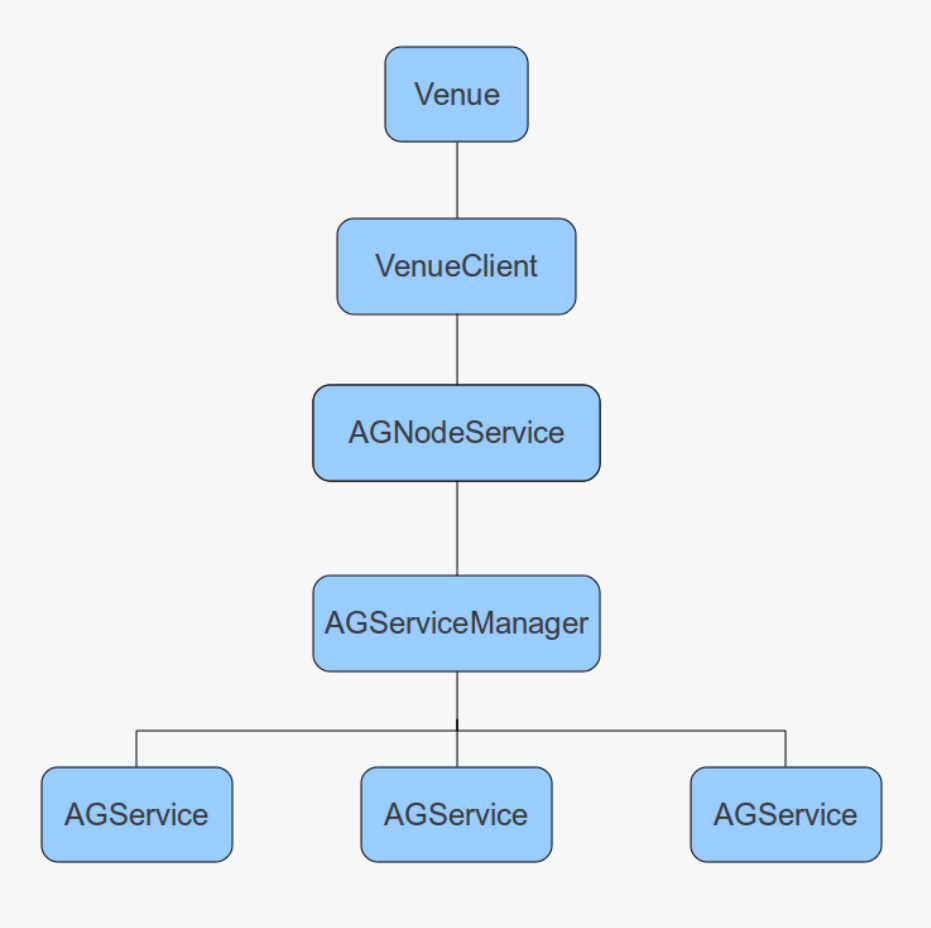


Figura 5.1: Arquitectura de los Componentes de Access Grid

Cada uno de estos componentes del diagrama tiene su respectiva interfaz de servicios web. *Access Grid* está diseñado en base a servicios web basado en SOAP. Para facilitar el desarrollo de *Access Grid*, todas las clases cliente han sido generadas y se han distribuido con el software. Estas clases simplifica, significativamente, la tarea de realizar métodos de llamadas remotas, así, en la mayoría de los casos, el desarrollador únicamente necesita instanciar un objeto con una dirección URL válida de un servicio web y comenzar a ejecutar sus métodos.

Los componentes mostrados en el diagrama anterior están descritos en el módulo de Python llamado `AccessGrid.interfaces`. Hay que destacar que cada uno de estos componentes, a la hora de importarlos en nuestra aplicación, sigue un mismo patrón que es el siguiente:

```
1     from AccessGrid.interfaces.COMPONENT_client import COMPONENTIW
```

Donde COMPONENT es uno de los componentes de servicios web de *Access Grid*.

Una vez explicado todo ésto, se pasará a mostrar una serie de ejemplos sencillos para entender mejor esta API.

Ejemplo 1: Mostrar todos los clientes de una Sala dada

En el primer ejemplo que se va a mostrar, nuestra aplicación accederá a una Sala dada cualquiera y, luego, mostrará por pantalla un listado de todos los perfiles de todos los clientes que estén conectados a dicha Sala.

Para ayudar a entender mejor la API del *Toolkit de Access Grid*, debemos pararnos a pensar qué clases vamos a necesitar, dependiendo del resultado que queremos obtener. Por ejemplo, en nuestro caso, queremos obtener un listado de todos los clientes conectados a una Sala. Parece entonces lógico pensar que la clase `Venue` (Sala en inglés) es la más idónea para utilizar, ya que, todos los clientes se conectan a una Sala, así que, parece lógico pensar que la clase `Venue` dispondrá de algún método que nos devuelva todos los clientes que contiene dicha Sala. Efectivamente, ésto es así, y lo podemos observar en el siguiente código:

```
# -*- coding: utf-8 -*-
#############################################################################
#                  INTRODUCCION A ACCESS GRID                               #
# Ejemplo 1 - Obtener un listado de Clientes conectados a una Sala dada #
#############################################################################

import agversion
agversion.select(3)

from AccessGrid.interfaces.Venue_client import VenueIW

venue = VenueIW('https://vv3.mcs.anl.gov:8000/Venues/default')

venueState = venue.GetState()
for client in venueState.clients:
    print client
```

Pasamos a explicar cada una de las instrucciones del ejemplo.

En primer lugar, como se comentó anteriormente, para la versión 3.x de *Access Grid*, hay que indicar qué versión del sistema queremos utilizar, ello se hace con las siguientes instrucciones:

```
    import agversion
    agversion.select(3)
```

Después, como se ha comentado en el inicio del ejemplo, se concluyó que la clase a utilizar sería la clase `Venue`, así pues, instanciamos un objeto de dicha clase. Indicar que, para ello, es necesario importar también su respectiva clase para poder utilizarla (recordar el patrón a seguir) y, por otro lado, en el constructor le pasamos la dirección URL válida de una Sala, en este caso, la sala por defecto del servidor de salas `vv3.mcs.anl.gov`:

```
        from AccessGrid.interfaces.Venue_client import VenueIW
        venue = VenueIW('https://vv3.mcs.anl.gov:8000/Venues/default')
```

Una vez obtenido el objeto de una Sala dada, obtener el listado de clientes es bastante simple, basta con obtener el estado de la Sala y, entre las propiedades, se incluye el listado de clientes conectados:

```
        venueState = venue.GetState()
```

```
        for client in venueState.clients:
        print client
```

Para ejecutar esta aplicación ejemplo basta con introducir en un Terminal o en una Consola la siguiente orden:

```
python ejemplo01.py
```

El resultado que veremos en la consola sería algo parecido a ésto:

```
Profile Type: user
Name: Jesus Cea
Email: jesus.cea@uca.es
Phone Number:
Location:
Venue Client URL: http://isar.mcs.anl.gov:11000/VenueClient
Public ID: c0a8fe0210d66704fe6bbf15d743c8ab
Home Venue: https://vv3.mcs.anl.gov:8000/Venues/default
```

Ejemplo 2: Mostrar un listado de todas las Salas que dispone un Servidor de Salas

Este ejemplo es muy similar al ejemplo anterior, salvo que, en este caso, en lugar de solicitar todos los clientes conectados a una Sala dada, se pide mostrar un listado de todas las Salas dado un Servidor de Salas.

Volviendo a insistir en pararse a pensar qué clase o clases se va a necesitar para solucionar este ejemplo, hay que hacer un razonamiento similar al ejemplo anterior. En este caso, si un Servidor de Salas está compuesto por varias Salas, es lógico pensar que la clase que podría facilitarnos el listado de todas las Salas puede ser la clase `VenueServer`. Efectivamente, dicha clase, nos ofrece la solución:

```
# -*- coding: utf-8 -*-
#####################################################################
#                  INTRODUCCION A ACCESS GRID                       #
# Ejemplo 2 - Obtener un listado de Salas dado un Servidor de Salas #
#####################################################################

import agversion
agversion.select(3)

from AccessGrid.interfaces.VenueServer_client import VenueServerIW

venueServerUrl = 'https://vv3.mcs.anl.gov:8000/VenueServer'

venueServer = VenueServerIW(venueServerUrl)

venueList = venueServer.GetVenues()
for venue in venueList:
    print venue.name
```

Podemos observar que, una vez declarado un objeto de tipo `VenueServerIW`, basta con llamar a la función `GetVenues()` para obtener todas las Salas de un Servidor de Salas.

Aclarar que la instrucción # coding: utf8 sirve para poder introducir en nuestro código fuente caracteres nacionales, como acentos o ñ (bien en nuestros comentarios o bien en una cadena de caracteres).

Ejemplo 3: Dada una Sala, introducir un cliente en dicha Sala

En este último ejemplo, se pretende realizar una aplicación en la que, dada una dirección URL de una Sala cualquiera, instanciar un objeto de tipo Sala (clase `Venue`) e introducir dicho objeto en la Sala.

```
# -*- coding: utf-8 -*-
#############################################################################
#                  INTRODUCCION A ACCESS GRID                               #
# Ejemplo 3 - Dada una direccion de Sala, introducir un cliente en ella #
#############################################################################

import agversion
agversion.select(3)

from AccessGrid.interfaces.VenueClient_client import VenueClientIW

venueClientUrl = 'http://hostname:11000/VenueClient'
venueUrl = 'https://vv3.mcs.anl.gov:8000/Venues/default'

venueClient = VenueClientIW(venueClientUrl)
venueClient.EnterVenue(venueUrl)
```

5.3.8. Conclusiones

Se puede observar que la API del *Toolkit de Access Grid* es trivial y muy poderoso. Los métodos proporcionados por esta API nos hace despreocuparnos de realizar todo el proceso para establecer comunicaciones y enviar mensajes al resto de clientes, gracias a los componentes de servicios web diseñados.

5.4. Corrección de Código

Una vez explicado unas nociones sobre cómo desarrollar para *Access Grid* utilizando su Toolkit, pasaremos a explicar en este y en los próximos apartados todo el desarrollo que se ha realizado para este proyecto. Concretamente, en este apartado se hablará sobre la corrección de código del sistema *Access Grid*, ya que, se han encontrado algunos errores, se han corregido y se han notificado al equipo de desarrollo de *Access Grid*.

A continuación se detallan los errores encontrados.

5.4.1. Compatibilidad con Windows 7

Errores de ejecución

Para hacer funcionar el sistema *Access Grid* se han encontrado varios problemas. En una instalación limpia de *Access Grid*, la primera vez que se ejecuta, una vez introducido nuestros datos para generar nuestro perfil de usuario, nos aparece la pantalla principal de *Access Grid* sin problemas:

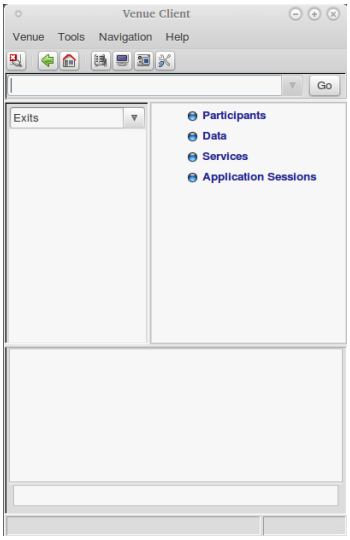


Figura 5.2: Pantalla Principal de Access Grid

Por lo que, a primera vista, parecía que el sistema *Access Grid* iba a funcionar sin problemas con el sistema operativo Windows 7. Sin embargo, cuando introducimos una dirección URL del Servidor de Salas que queremos acceder, el sistema realiza la conexión correctamente, comienza a cargar la configuración de nuestro perfil (número de cámaras y calidad, más otros servicios), pero, de repente, en mitad de esta carga, aparece un error de que Python dejó de funcionar, obligándonos a cerrar la aplicación:

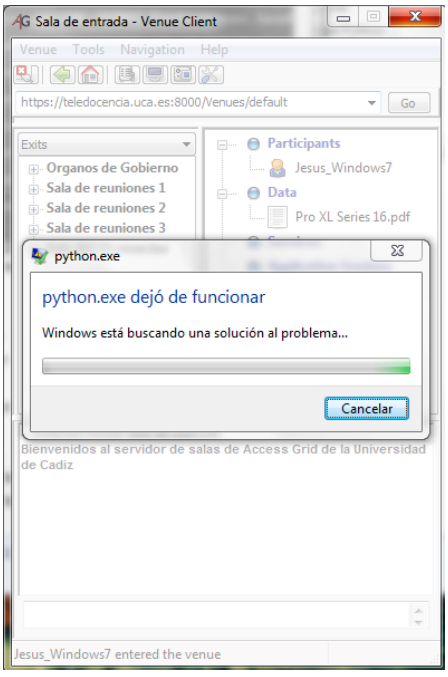


Figura 5.3: Primer error de Access Grid en Windows 7

Sin embargo, en la segunda ejecución, aparece un nuevo problema, diferente al primero. La segunda (y consecutivas) ejecuciones del programa, no llega a ejecutar la pantalla principal de *Access Grid*,

mientras sale el logo de *Access Grid*, donde se va inicializando la aplicación, sin mostrar ningún tipo de error, la aplicación se cierra instantáneamente.

Access Grid incluye un modo de ejecución, permitiendo activar un depurador (recordar en el apartado anterior los argumentos estándares). Ésto permite ejecutar el cliente de *Access Grid* además de una consola donde se irá notificando de todos los sucesos que van ocurriendo a lo largo de la ejecución del programa. Así, si se provocan los errores anteriormente comentados, en la consola de depuración puede que indique el motivo del error (también se pueden observar los ficheros logs generados).

Así pues, desinstalando completamente la aplicación (para poder provocar el primer error), y volviendo a realizar una instalación limpia, realizamos los mismos pasos que antes. Observando la consola, no muestra ningún tipo de error, pero sí las últimas líneas que se han ejecutado:

```
1           log.info("VenueClient.StartBeacon: Address %s/%s"
2           %(self.beaconLocation.host, self.beaconLocation.port))
3           self.beacon.Start()
```

Este par de instrucciones por un lado, muestra en el depurador (y también se almacena en el fichero log) que la herramienta *Beacon* va a comenzar con una determinada dirección URL (supuestamente, la dirección del Servidor de Salas) y un puerto. Luego, invoca la ejecución de *Beacon*.

Beacon es una herramienta muy útil que nos ayuda a depurar problemas con las conexiones multicast. Nos proporciona información muy útil sobre la actividad existente en una conexión de un grupo multicast. Nos indica si todo funciona correctamente o, por el contrario, si existe algún tipo de problema y cuál es el problema.

Para verificar si el problema se encuentran en esas últimas líneas de código, se decide comentar dichas líneas y repetir el proceso (previa reinstalación completa del sistema para volver a provocar dicho error), y se observa que la aplicación funciona correctamente. Así pues, o bien el problema está en que las variables `self.beaconLocation.host` y `self.beaconLocation.port` no poseen ningún valor, o bien, existe algún problema con inicializar *Beacon* bajo Windows 7.

Depurando un poco más la aplicación, se intenta mostrar los valores de ambas variables y se observa que dichas variables contienen valores correctos (de hecho, el fichero log almacena correctamente la información que se muestra en esa instrucción). Por lo tanto, el problema radica en la instrucción `self.beacon.Start()`. Decidimos comentar esta línea y se comprueba que el sistema funciona correctamente.

Pasamos al segundo error, repetimos el proceso anterior (ejecutar *Access Grid* en modo depuración) y se observa que, cuando se cierra la aplicación, la consola muestra el siguiente mensaje:

```
  File "C:\Python24\lib\site-packages\AccessGrid3\AccessGrid\VenueClient.py", line 266, in __init__
    if int(self.preferences.GetPreference(Preferences.MULTICAST)):
ValueError: invalid literal for int(): False
```

Según el compilador, indica que la conversión a entero era incorrecta. Nos dirigimos a esa línea del fichero "VenueClient.py" y observamos ese trozo de código:

```
1           if int(self.preferences.GetPreference(Preferences.MULTICAST)):
2           self.transport = "multicast"
3       else:
```

```
4           self.transport = "unicast"
```

Básicamente, este conjunto de instrucciones cogen la configuración almacenada en el sistema y comprueba si está activado o no la opción de utilizar la red multicast. Se decide cambiar dicho *int* por *bool*, es decir, quedaría así:

```
1           if bool(self.preferences.GetPreference(Preferences.MULTICAST)):
2           self.transport = "multicast"
3       else:
4           self.transport = "unicast"
```

Volvemos a ejecutar la aplicación y se observa que ahora se ejecuta correctamente sin problemas. Con estas correcciones, se ha conseguido ejecutar *Access Grid* en Windows 7.

La ventana de preferencias no se muestra

Otro error encontrado en *Access Grid* ejecutándose bajo el sistema operativo Windows 7 es que, cuando intentábamos entrar en la configuración de la aplicación, dirigiéndonos en la barra de herramientas a "Tools− >Preferences...", no se muestra dicha ventana de configuración.

Como se ha comentado en el error anterior, para ayudar a encontrar la solución de este problema, se recomienda ejecutar *Access Grid* en modo de depuración. Así pues, ejecutamos *Access Grid* en modo depuración y repetimos los pasos anteriores para ver si la consola muestra algún tipo de error, es decir, ejecutamos la aplicación y nos dirigimos a "Tools− >Preferences...". La consola muestra el siguiente error:

```
  File "C:\Python24\lib\site-packages\AccessGrid3\AccessGrid\PreferencesUI.py",line 1173, in __init__
    self.multicastButton.SetValue(not int(preferences.GetPreference(Preferences.
MULTICAST)))
ValueError: invalid literal for int(): False
```

Vemos que el error es muy similar al segundo error encontrado en el problema anterior. Abrimos el fichero "PreferencesUI.py" y nos dirigimos a la línea donde aparece el error (línea 1173), el código es el siguiente:

```
1           self.multicastButton.SetValue(not int(preferences.GetPreference(
               Preferences.MULTICAST)))
```

Una vez más vemos que Python no es capaz de hacer una conversión interna de entero a booleano, por lo que se decide cambiar *int* por *bool*, quedando la instrucción así:

```
1           self.multicastButton.SetValue(not bool(preferences.GetPreference(
               Preferences.MULTICAST)))
```

Almacenamos los cambios y observamos que ahora sí se muestra la ventana de preferencias correctamente.

Resumiendo

De forma resumida, para que *Access Grid* funcione correctamente en Windows 7 [9], hay que hacer lo siguiente:

[9]Windows 7 es una versión renovada de Windows Vista, así que, aunque no se ha comprobado, suponemos que estos cambios también afectarán a los sistemas Windows Vista

- Editar (con cualquier editor de textos) el fichero VenueClient.py, alojado en `C:\Python24\¬ Lib\site-packages\AccessGrid3\AccessGrid`.

- Sustituir, en la línea 266:

```
if int(self.preferences.GetPreference(Preferences.MULTICAST)):
    self.transport = "multicast"
else:
    self.transport = "unicast"
```

Por:

```
if bool(self.preferences.GetPreference(Preferences.MULTICAST)):
    self.transport = "multicast"
else:
    self.transport = "unicast"
```

- Comentar (se comenta con el símbolo '#') la línea 1208.

- Para que funcione la ventana "Preferences..." del menú "Tools", editar (con cualquier editor de textos) el fichero "PreferencesUI.py", alojado en `C:\Python24\Lib\site-packages\¬ AccessGrid3\AccessGrid`. Sustituir, en la línea 1173:

```
self.multicastButton.SetValue(not int(preferences.GetPreference(
    Preferences.MULTICAST)))
```

Por:

```
self.multicastButton.SetValue(not bool(preferences.GetPreference(
    Preferences.MULTICAST)))
```

- Ejecutar el Cliente de *Access Grid*.

Un método más sencillo de solucionar el error Beacon

Los errores encontrados fueron notificados al equipo de desarrollo de *Access Grid*. Aunque se mostraron muy agradecidos por los errores comentados, nos comentaron que, efectivamente, existe un problema de incompatibilidad entre la herramienta *Beacon* y el sistema *Windows 7*. De momento no saben cómo solucionar este problema, sin embargo, en lugar de corregir el código fuente, comentando, y evitando, la ejecución de *Beacon*, existe un método mucho más rápido y más directo, como solución temporal.

- Ejecutamos el sistema *Access Grid* pero no lo conectamos a ningún Servidor de Salas.

- Nos dirigimos a "Tools− >Preferences..." :

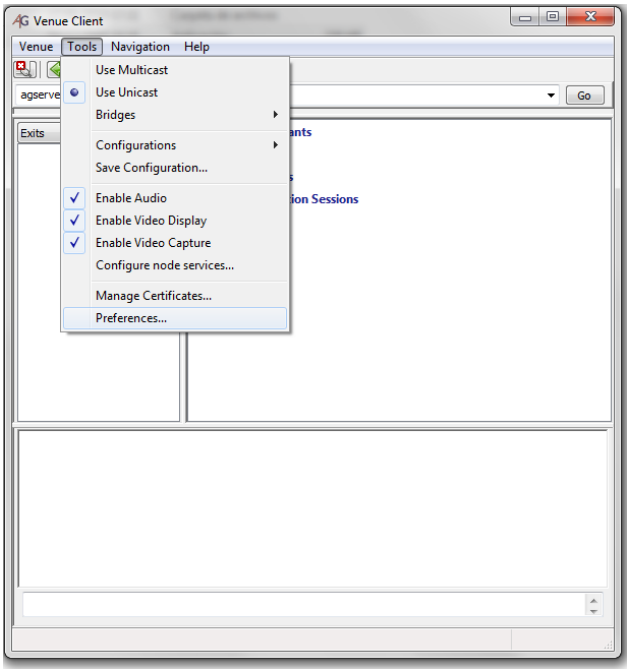


Figura 5.4: Dirigiéndonos a la configuración de Access Grid

- Nos dirigimos a la sección **Network**

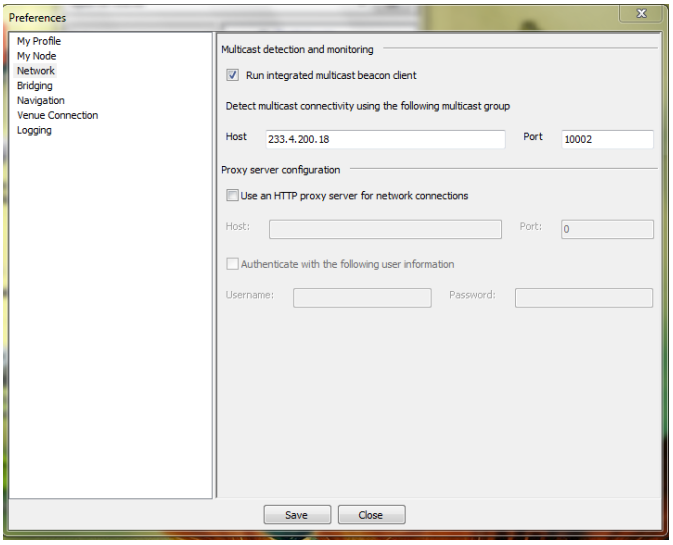


Figura 5.5: Dirigiéndonos a la configuración de Access Grid

- Desactivamos la casilla **`Run integrated multicast beacon client`**

Figura 5.6: Dirigiéndonos a la configuración de Access Grid

- Pulsamos el botón **Save** para guardar los cambios.

De esta manera, desactivamos que se ejecute, en cualquier parte del programa, la herramienta *Beacon* y no nos aparecerá más el error.

5.4.2. Problemas con los certificados digitales

Otro error encontrado, a la hora de utilizar *Access Grid*, sucede cuando queremos utilizar certificados digitales, normalmente queremos hacer uso de estos certificados cuando queremos controlar el acceso a una Sala de *Access Grid*.

De forma resumida, si se desea controlar el acceso a una Sala, o incluso al propio Servidor de Salas por completo, se debe configurar la aplicación *VenueManager* que es la que nos permite gestionar el Servidor de Salas, pudiendo crear nuevas Salas, editar o borrar las ya existentes y, ademas, controlar el acceso a ellas.

Como se ha comentado, *Access Grid* controla el acceso a través de los certificados digitales [10]. De forma resumida, un certificado digital es un documento digital donde una entidad certificadora garantiza la identificación del usuario a través de dicho documento. Existen varios tipos de certificados digitales, sin embargo, los que usaremos en Access Grid son los certificados **x509** [11].

La forma de hacer referencia a un certificado digital es a través de su **Distinguished Name** (DN). La autoridad certificadora (AC) emite el certificado una clave pública a un DN. El DN es la identificación única de un certificado, se compone de varios atributos. La forma de un DN es una secuencia de pares atributos=valor separados por coma o por un espacio y el orden en el que aparecen los pares es importante. Estos atributos reciben el nombre de **Relative Distinguished Names** (RDN). Un ejemplo de DN podría ser el siguiente:

```
/O=Access Grid/OU=agdev-ca.mcs.anl.gov/OU=agserver.uca.es/CN=Jesus Cea Oliva
```

Así pues, a partir de un DN, ***Access Grid* puede controlar el acceso de un usuario** a una Sala concreta o al Servidor de Salas en sí.

[10] http://es.wikipedia.org/wiki/Certificado_digital
[11] http://es.wikipedia.org/wiki/X.509

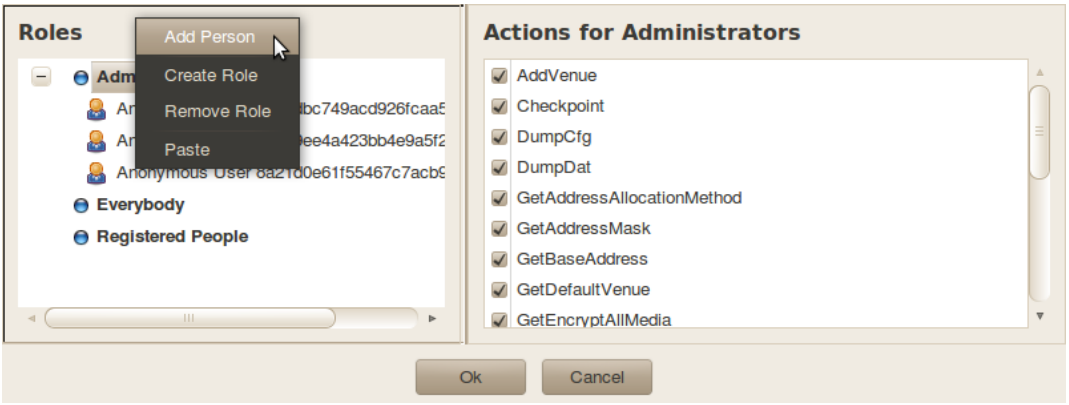


Figura 5.7: Ejemplo de cómo asignar a un usuario (Jesús, en este caso) como Administrador I

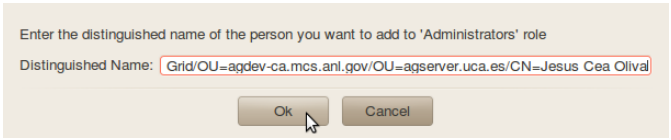


Figura 5.8: Ejemplo de cómo asignar a un usuario (Jesús, en este caso) como Administrador II

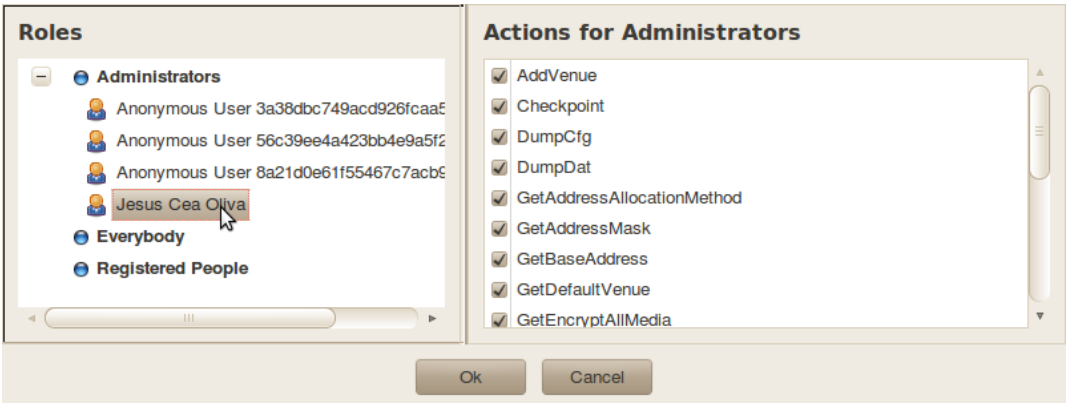


Figura 5.9: Ejemplo de cómo asignar a un usuario (Jesús, en este caso) como Administrador III

Puede encontrar información mucho más detallada en el Apéndice, donde se incluye un manual completo del sistema *Access Grid*.

Una vez explicado, de forma resumida, cómo controlar el acceso a un usuario en una Sala de *Access Grid*, será más fácil entender el error encontrado. Cuando, en un Servidor de Salas, se da acceso a una Sala concreta a un usuario, a través de su certificado digital, dicho certificado digital debe estar instalado el cliente en aquel PC que vaya a utilizar para conectarse a *Access Grid* (una vez más, en el manual adjunto en los apéndices se detallan estos pasos).

El problema surge cuando, el cliente, una vez dado acceso a la Sala y teniendo su certificado instalado, decide acceder a dicha Sala. Aparece un mensaje de error como el que se muestra a continuación:

```
    File "/usr/lib/python2.6/_strptime.py", line 325, in _strptime_time (data_string, format)
  ValueError: time data 'Apr 21 00:52:18 2007 GMT' does not match format ' %b %d %H: %M: %S %Y %Z'
```

Se observa un error similar, a la hora de ejecutar el Gestor de Certificados, *Certificate Management*, ya que el programa, simplemente, no se ejecuta. Como se ha comentado varias veces anteriormente, se ejecuta la aplicación de nuevo, activando la depuración del programa y, se observa en la consola que las últimas líneas ejecutadas muestran el mismo error que el que se ha mostrado anteriormente.

Aclarar que este error **sucede en máquinas Linux**, pero en Windows funciona correctamente. Solucionar este problema fue bastante complicado, ya que, como se puede observar, no se sabe qué fichero realiza esta acción y, por tanto, produce este error, ya que, el fichero que produce este error es un fichero estándar de *Python*, es decir, hay un fichero de *Access Grid* que utiliza esta función y, por tanto, produce su error.

Se miraron los ficheros que se podrían considerar como ”principales”, en el sentido de que, son donde, a priori, puede aparecer este error, según la ejecución hecha. Es decir, el error se produce al entrar en una Sala ejecutando el Cliente, es obvio pensar que dicho error puede estar en el código fuente del cliente de *Access Grid*, es decir, los ficheros `Venueclient.py` y `VenueClientUI.py`. Sin embargo, en estos ficheros no se encontró nada relacionado con los certificados que pudiera indicar que el error se encontrase ahí.

Otros ficheros que se observaron fueron los del Servidor de Salas (`VenueServer.py`) y el Gestor de Certificados (`CertificateManagement.py`), pero no se encontró nada.

Finalmente se hizo una traza, gracias a los ficheros Logs generados. El fichero `Utilities.py` es el último fichero que se ejecutó y es éste quien llamó a la función _strptime con la instrucción:

```
1     before_tuple = time.strptime(str(before_time), format)
```

dentro de la función `IsExpired(x509)`. Sin embargo, se observó los valores de las variables `before_time` y `format` y ambas contenían un valor aceptable, no debía de dar ningún tipo de error.

Por lo tanto, se pensó que el problema podría radicar en que el sistema *Access Grid* tuviera algún tipo de configuración de idioma y éste estuviera establecido al sistema extranjero, mientras que, el formato de los certificados españoles, difieren de éstos y, por ello, salta el error. Habría que buscar qué fichero del sistema *Access Grid* es quien establece la codificación de caracteres del sistema y de los certificados.

Volviendo a realizar la traza de estas instrucciones, se observa que la raíz de todo comienza en el fichero `CertificateManager.py`, que es quien llama a estas funciones para validar los certificados (si han expirado, etc). Por cada función que contiene este fichero, se observa que, aquellas funciones que tienen relación con los certificados digitales, como por ejemplo, `ImportRequestedCertificate` o `ImportCACertificates`, siempre importa un módulo *Access Grid*, llamado `Toolkit` y, siempre lo llaman para obtener valores ”generales”, como por ejemplo, obtener la ruta de la aplicación (con `GetDefaultApplication()`, así que se decide observar este fichero, a ver si contiene alguna instrucción relevante.

Efectivamente, al abrir este fichero, se observa que, entre otros módulos, se importa el **módulo `os`** [12]. Este módulo, entre otras funciones, nos permite establecer variables de entorno y configurar el sistema

[12] `http://docs.python.org/library/os.html`

a un estado que a nosotros, como desarrolladores nos interese. Mirando dicho fichero, encontramos la instrucción culpable de todo:

```
os.environ['LANG'] = "C"
```

Con esta instrucción estamos indicando a la aplicación que establezca el sistema a un lenguaje determinado. Buscando información por la red, parece ser que esta instrucción es correcta, sin embargo, parece que *Access Grid* hace caso omiso de ella y no aplica los cambios solicitados.

Siguiendo buscando información por la red, vemos que existe un módulo llamado **módulo `locale`** [13]. Este módulo permite a nuestra aplicación "adaptarse" a la localización concreta del país donde esté ejecutándose. Por ejemplo, si en una PC español introducimos esta instrucción

```
locale.setlocale(locale.LC_ALL, locale.getdefaultlocale())
```

El sistema nos devolvería el siguiente resultado:

```
'es_ES.UTF8'
```

Indicándonos que la localización por defecto de nuestro sistema es el español UTF8. Si ahora escribimos:

```
hoy = date.today()
hoy.strftime("%m-%d-%y. %d %b %Y es %A. hoy es %d de %B.")
```

El sistema nos devolvería el siguiente resultado:

```
'07-19-09. 19 jul 2009 es domingo. hoy es 19 de julio.'
```

Sin embargo, si establecemos la localización a otro país, el resultado sería diferente.

Así pues, añadimos, justo después de esta instrucción, la siguiente instrucción:

```
locale.setlocale(locale.LC_ALL,'C')
```

Guardamos los cambios y repetimos el proceso inicial para producir el error. Efectivamente vemos que ahora, todo funciona correctamente gracias a este cambio.

Notificación del error a los desarrolladores

Una vez encontrada la solución, decidimos ponernos en contacto con los desarrolladores de *Access Grid* para comunicarles el error y la posible solución que se ha encontrado. Al principio no eran capaces de producir nuestro error, hasta que establecieron como idioma predeterminado de su sistema a Español.

A partir de dicho cambio, sí que fueron capaces de replicar nuestro error encontrado. Nos comunicaron que, aunque la solución aportada era buena y funcional, realmente no era una solución aceptable, existía un fallo de implementación del sistema y que, en próximas versiones, este error será subsanado. El equipo de desarrollo de *Access Grid* estuvieron muy agradecidos por haber notificado este error.

[13] `http://docs.python.org/library/locale.html`

5.4.3. Problemas a la hora de introducir valores manuales de direccionamiento en el Servidor de Salas

Siguiendo con el uso y aprendizaje del sistema *Access Grid* se observa que se produce un nuevo error. En este caso el error se producía en la aplicación del Gestor de Salas, es decir, la aplicación *VenueManagement*.

De forma resumida, ya que toda la información referente al uso del sistema *Access Grid* lo encontrará en un apéndice de este documento, la aplicación *VenueManagement*, además de otras funciones, como el control de acceso en una sala (como se comento en el error anterior), permite configurar una Sala. Por ejemplo, se puede dar un nombre a esa Sala, también se puede indicar qué salidas tiene la Sala (es decir, a qué Salas se pueden acceder a través de esta Sala), etc. Pero, además, también permite configurar el propio Servidor de Salas, una vez más, se puede controlar el acceso al Servidor de Salas, se puede indicar que encripte todo el flujo de datos entre Cliente y Salas y, además, permite asignar un direccionamiento de red multicast tanto dinámico como estático:

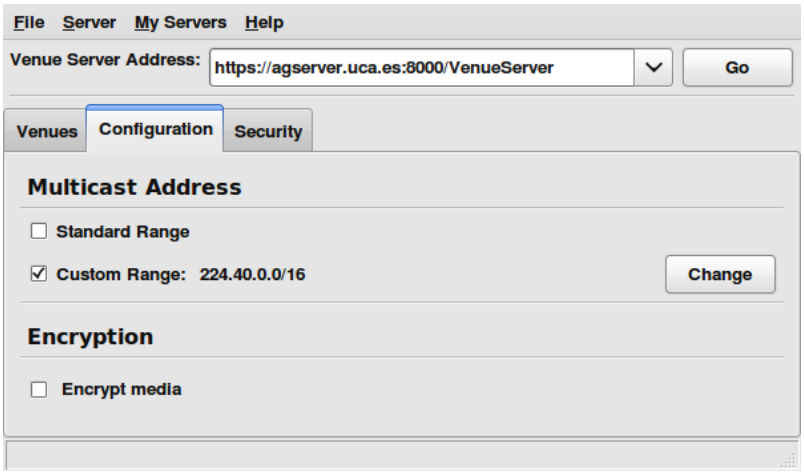


Figura 5.10: Estableciendo una dirección de red multicast estática

El problema se produce cuando se desea introducir unos valores de una dirección de red estática, para configurar el Servidor de Salas de forma que, por cada Sala que se vaya creando, se le asigne una IP estática dentro de un grupo indicado. El problema en sí es que, una vez aparece la pantalla para introducir la dirección IP, no se puede escribir nada, tanto texto, como números. Curiosamente, si copiamos la dirección en cualquier otro lugar, como por ejemplo, en un editor de texto, y se va copiando cada trozo de la IP, luego sí se puede pegar ese trozo copiado en la ventana (usando la combinación de teclas Control + V).

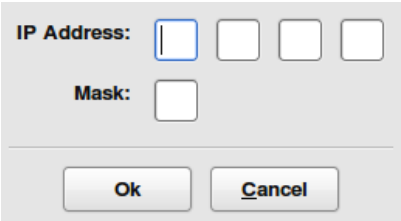


Figura 5.11: En esta ventana no podemos introducir ningún carácter

En esta ocasión, resultó algo más fácil localizar el origen del error. Supuestamente, la aplicación está diseñada de forma que, a la hora de introducir valores en esta ventana, únicamente permita introducir números, ya que una dirección IP, está formada por cuatro bloque de números de tres cifras, separados por un carácter punto (dicho carácter no hace falta introducirlo puesto que la ventana ofrece cuatro campos por separado). Así pues, se supuso que debe existir un evento, tipo *OnKeyPress* o similar, de tal forma que dicho evento se debe activar a la hora de pulsar una tecla, se analizaría dicha tecla pulsada y, dependiendo de si es un número o no, se introduce o no el carácter en el campo correspondiente.

Como la aplicación que se está utilizando es el Gestor de Salas, es decir, la aplicación *VenueManagement*, se opta por mirar el código de esta aplicación.

Efectivamente, como esta aplicación (al igual que todas las aplicaciones que forman el sistema *Access Grid*) está desarrollada bajo *wxWidget*, se busca el evento carácter de dicho lenguaje. En este caso, se busca por la cadena `wx.EVT_CHAR`. Se encuentra una clase llamada, `DigitValidator`, vemos que su nombre concuerda con la suposición que se ha hecho, esta clase, entre otras funciones, valida si la tecla pulsada se trata de un carácter o no.

Pues bien, la instrucción en concreto de esta función es la siguiente:

```
1      def __init__(self, flag):
2          wx.PyValidator.__init__(self)
3          self.flag = flag
4          wx.EVT_CHAR(self, self.OnChar) # Llama a una funcion llamada OnChar
```

Como se puede observar en el código, el evento carácter llama a una función *OnChar* para verificar el carácter introducido. Nos dirigimos a dicha función, dentro del mismo fichero y se destaca lo siguiente:

```
1      def OnChar(self, event):
2          key = event.KeyCode()
3
4          if key < wx.WXK_SPACE or key == wx.WXK_DELETE or key > 255:
5              event.Skip()
6              return
7
8          (...)
9          # Otras comprobaciones
```

La instrucción `key = event.KeyCode()` no es correcta. Si observamos la documentación de *wxPython*, concretamente, con respecto a obtener la tecla pulsada, recomiendan usar la funcion `GetKeyCode()`,`http://www.wxpython.org/docs/api/wx.KeyEvent-class.html#KeyCode`.

Por lo tanto, si cambiamos la instrucción:

```
1      def OnChar(self, event):
2          key = event.KeyCode() # Esto no es correcto
```

Por:

```
1      def OnChar(self, event):
2          key = event.GetKeyCode() # Esto si es correcto
```

La aplicación, ahora, debería de permitirnos dejar introducir números (y únicamente números) en los respectivos campos y, por tanto, poder introducir una dirección de red multicast estática.

Notificación del error a los desarrolladores

Una vez encontrada la solución, decidimos ponernos en contacto con los desarrolladores de *Access Grid* para comunicarles el error y la posible solución que se ha encontrado. Al principio no eran capaces de producir nuestro error, hasta que establecieron como idioma predeterminado de su sistema a Español, una vez más.

A partir de dicho cambio, sí que fueron capaces de replicar nuestro error encontrado. El equipo de desarrollo de *Access Grid* estuvieron muy agradecidos por haber notificado este error y su solución, y en futuras versiones subsanarán el error.

5.4.4. Problemas de compatibilidad con los nuevos sistemas Linux

Un último error encontrado, y bastante grave, se encontró a lo largo de todo el proceso de aprendizaje y uso de *Access Grid*. Sin embargo, hasta pasados varios meses, no éramos conscientes de que se trataba de un error del sistema y se atribuía a que hacíamos un mal uso de él.

Síntomas encontrados

A continuación se detalla el conjunto de problemas encontrados y, después de varias conversaciones con los desarrolladores del sistema *Access Grid*, se concluyó que el origen del problema era el mismo.

No se puede compartir archivos

Uno de los primeros problemas que se tuvo, a lo largo del aprendizaje y uso de *Access Grid*, fue el poder compartir archivos entre los participantes conectados. Compartir un archivo en *Access Grid* es sumamente fácil. En la interfaz principal del programa, se encuentra dividida en varios bloques, uno de ellos se llama **Data**. En este bloque se encuentra el conjunto de directorios y ficheros que los usuarios suben al Servidor de Salas para compartirlo con el resto de participantes.

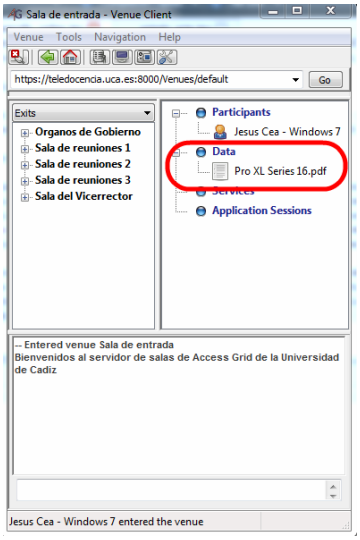


Figura 5.12: Bloque donde se encuentran los ficheros compartidos

Si uno desea compartir un archivo, o directorio, con el resto de participantes, basta con dirigirse a la barra de herramientas a ”Venue− >Add Data...”, para compartir un archivo, o ”Venue− >Add Directory...”, para compartir un directorio. También puede repetir este proceso pulsando el segundo botón del ratón, en el bloque ”Data” de la interfaz principal de *Access Grid*.

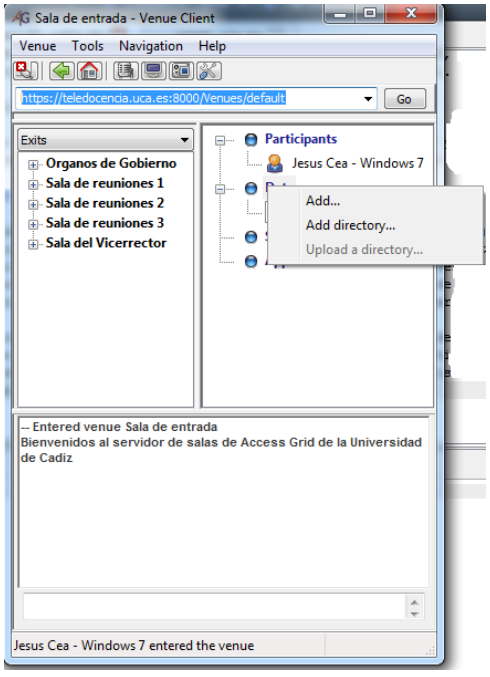


Figura 5.13: Compartir archivos entre participantes

El problema encontrado surge a la hora de compartir archivos, puesto que, cuando un cliente envía un archivo a compartir, en la parte inferior de la interfaz principal del cliente de *Access Grid*, aparece una barra de estado, que incluye una barra de progreso, indicando el progreso actual que lleva el programa subiendo el archivo. El problema es que, nunca termina, cuando la barra de progreso llega al final, no

desaparece, se queda congelado y, si se finaliza la sesión, el fichero no se ha compartido. En definitiva, **se bloquea la compartición de archivos**.

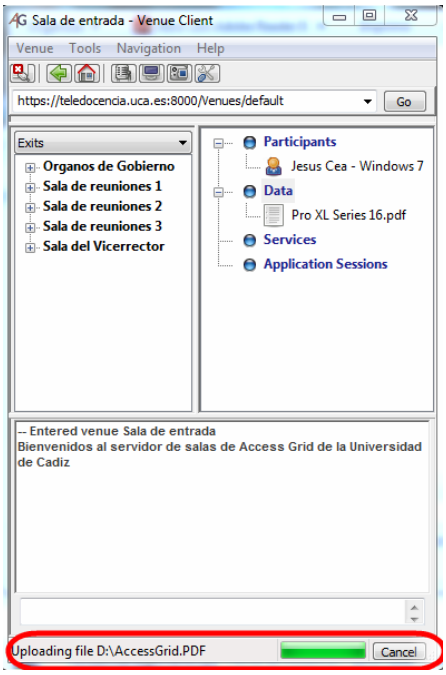


Figura 5.14: El proceso nunca acaba y al final no se comparte el archivo

Los eventos no se llevan a cabo

Otro problema grave encontrado tiene relación con las Aplicaciones Compartidas. En la próxima sección se detallará sobre las Aplicaciones Compartidas. Básicamente, una aplicación compartida es una aplicación integrada en el sistema *Access Grid*, cuya característica es que los cambios que un cliente haga en esa aplicación afecta, de alguna manera, al resto de participantes invitados a utilizar la aplicación. Por ejemplo, si la aplicación es una pizarra compartida, cuando un usuario dibuje sobre ella, el resto de participantes invitados a dicha aplicación, visualizarán lo que el usuario dibujó e incluso podrán modificar el dibujo, añadiendo más dibujo o borrando el suyo.

Para que esto se lleve a cabo, la API del *Toolkit de Access Grid*, incorpora las herramientas necesarias para poder crear eventos compartidos, de forma que si el desarrollador desea transmitir algún cambio realizado en la aplicación, al resto de participantes, deberá crear un evento compartido. Así, el sistema *Access Grid*, es notificado de que un cierto evento ha ocurrido y desea transmitirlo al resto de usuarios conectados.

El problema encontrado surge a raíz de que, tanto a la hora de utilizar aplicaciones compartidas, como a la hora de desarrollar las nuestras propias, los eventos no se enviaban al resto de usuarios, quedándose las aplicaciones colgadas (o funcionando de manera local). Por ejemplo, una de las aplicaciones compartidas que se usó fue el visor compartido de presentaciones Power Point. Cuando un usuario cambia de transparencia, al resto de usuarios conectados también se le cambian la transparencia. Sin embargo, debido a este error, el usuario que cambiaba la transparencia podía cambiarla sin problemas, pero dicho cambio, no se notificaba al resto de participantes, quedándose estancados en la transparencia anterior. Lo mismo sucedía con el resto de aplicaciones compartidas.

Motivos de los errores

Aunque no se sabía cómo solucionar estos problemas, después de varias pruebas y varios escenarios de prueba llegamos a la conclusión de que estos problemas no surgían en todos los escenarios. Concretamente, en las máquinas Windows el sistema funcionaba correctamente, pudiendo compartir archivos y utilizar aplicaciones compartidas sin problema alguno. Concretando aún más, **si el Servidor de Salas está en una máquina Windows, no habrá ningún tipo de problema**, ya que, nos dimos cuenta de que los problemas los tenía la aplicación *VenueServer*, es decir, el Servidor de Salas.

Así pues, se realizaron aún más pruebas y más escenarios de pruebas, pero esta vez, probando diferentes distribuciones de Linux. Se observó que **el sistema *Access Grid* tenía problemas con las últimas versiones de cualquier distribución Linux**. Es decir, en una máquina con *Fedora 10* no había ningún tipo de problema en compartir archivos ni utilizando aplicaciones compartidas, en cambio, si usábamos *Fedora 12* volvían los problemas. Lo mismo ocurría con otras distribuciones probadas: *Ubuntu*, *SuSe* y *RedHat*.

Decidimos ponernos en contacto de nuevo con los desarrolladores de *Access Grid*. Se les explicó todos los escenarios de prueba realizados y los resultados que se han ido obteniendo. En un principio, no eran capaces de replicar nuestros problemas. El motivo era que realizaban las pruebas con distribuciones antiguas de Linux pero no realizaban las pruebas con las distribuciones que les habíamos indicado. Se hizo bastante hincapié en que realizasen las pruebas con las distribuciones que se habían probado y, efectivamente, fueron capaces de producir los problemas comentados.

Después de muchísimas pruebas, y después de haber mirado cientos de ficheros logs generados y compartidos entre los participantes en el listado de correo, se concluyó que ***Access Grid* (concretamente el Servidor de Salas) no funciona correctamente en las últimas distribuciones de Linux, debido a las nuevas versiones de *Python*** [14]. *Access Grid* está desarrollado bajo *Python 2.4*, por entonces, no existía ninguna librería para utilizar SSL ni SOAP, así que el equipo de desarrollo utilizaron los paquetes *m2crypto* y *ZSI*. Sin embargo, las nuevas versiones de Python, integran nuevas librerías y nuevos módulos soportando el uso de SSL.

Así que, se concluye que, estas nuevas librerías incluidas en las nuevas versiones de *Python* estén interfiriendo con los paquetes *m2crypto* y/o *ZSI*.

Error aún sin resolver

Aunque se ha determinado el posible motivo del error, el equipo de desarrollo de *Access Grid* están muy agradecidos por haber encontrado este error grave en el sistema. Sin embargo, nos comunicaron que solucionar este error iba a llevar bastante tiempo, ya que, en principio, no se sabe cómo solucionarlo.

Así que, a la espera de una nueva versión de *Access Grid*, donde el error encontrado se haya solucionado completamente, nos recomiendan que instalemos el sistema *Access Grid* bien en una vieja distribución del sistema Linux que se esté utilizando actualmente o, como último remedio, instalar *Access Grid* en una máquina Windows.

Parche temporal

Cuando se encontró este problema teníamos a nuestra disposición tanto el servidor de pruebas, cuyo sistema era ***Ubuntu Karmic (9.10)***, como el Servidor migrado desde el Campus de Cádiz hasta el Cam-

[14] Actualmente la versión de Python que se instala con Linux es la versión 2.6

pus de Puerto Real, cuyo sistema Linux era la última versión de *Ubuntu*, es decir, ***Ubuntu Lucid (10.04)***.

Los desarrolladores de *Access Grid*, concretamente **Christoph Willing**, quien le debemos agradecerle lo mucho que nos ha ayudado día a día, nos aportaron una pequeña solución para, de momento, hacer funcionar el Servidor de Salas en nuestras máquinas Linux.

Dicho parche consistía en instalar, en nuestras máquinas, la versión de *Python* más baja que nos dejara el sistema operativo y, luego, configurar el sistema para que utilice dicha versión como versión predeterminada de *Python*.

En *Ubuntu Karmic* se instaló, como versión más baja permitida, la versión *Python 2.5*. Sin embargo, **en *Ubuntu Lucid* no se podía instalar una versión más antigua de Python**, obligándonos a utilizar la versión 2.6.

Así pues, el Servidor de Salas que está bajo *Ubuntu Lucid*, a día de hoy, **no se puede solucionar** a la espera de una nueva versión de *Access Grid* o de que los desarrolladores aporten una solución para este sistema.

En cambio, para *Ubuntu Karmic* sí que fue posible solventar el error, aunque con matices. Los pasos a seguir son los siguientes:

- Escribir en una terminal el siguiente comando:

```
sudo apt-get install python2.5
```

- Configurar el sistema para que utilice Python 2.5 por defecto:

```
sudo update-alternatives --install /usr/bin/python python
     /usr/ bin/python2.5 20
sudo update-alternatives --install /usr/bin/python python
     /usr/ bin/python2.6 10
```

Una vez realizado estos pasos, podremos ejecutar el Servidor de Salas y comprobar que todo funciona correctamente.

Sin embargo, como se ha comentado, funciona todo correctamente, pero **con matices**, ya que, al ser la versión de *Python* por defecto la versión 2.6, las aplicaciones que estén desarrolladas o dependan de *Python* también están desarrolladas para dicha versión. Por lo tanto, si el servidor, además de utilizarlo como Servidor de Salas, se va a utilizar para otras aplicaciones, pueden surgir problemas por cambiar la versión por defecto de *Python*. Concretamente, un ejemplo directo, si dejamos como versión predeterminada de *Python* la versión 2.5, **tendremos problemas a la hora de actualizar el sistema**. En ese caso, tendremos que repetir el segundo paso anterior, asignándole un valor más alto a la versión 2.6, para que sea la versión de *Python* por defecto, realizar las tareas que se deseen realizar y, por último, volver al estado anterior. Es decir, tendríamos que hacer:

```
sudo update-alternatives --install /usr/bin/python python
    /usr/ bin/python2.6 30 # Cualquier numero mayor que 20
```

realizar luego las operaciones que sean necesarias con dicha versión de Python (actualizar el sistema, utilizar una aplicación, etc.) y luego volver a poner, como versión por defecto, a *Python 2.5*:

```
sudo update-alternatives --install /usr/bin/python python
    /usr/ bin/python2.6 10 # Cualquier numero menor que 20
```

5.4.5. Resumen de errores encontrados

De forma resumida, los errores que se han encontrado en el sistema *Access Grid* se encuentran agrupados en la siguiente tabla:

Error	**Sistema/s Operativo**	**Solucionado**	**Notificado**
Compatibilidad con Windows Vista / 7	Windows Vista / 7	Sí (Desactivando Beacon)	Sí
Error en el formato fecha de los certificados digitales	Linux / Windows	Sí	Sí
Error al establecer direccionamiento estático en el Servidor de Salas	Linux	Sí	Sí
Problemas de compatibilidad con nuevas versiones de Python	Linux	Sólo hasta versiones de Python inferior a 2.5	Sí

Tabla 5.1: Resumen de errores encontrados en *Access Grid*

5.5. Desarrollo de Aplicaciones Compartidas (Shared Applications)

El *Toolkit de Access Grid* proporciona un conjunto de librerías, módulos y tipos de datos para poder modificar o añadir nuevos elementos dentro del propio sistema *Access Grid*. Pero, además, la API del *Toolkit de Access Grid* proporciona un conjunto de librerías para desarrollar, lo que se denomina, **Aplicaciones Compartidas (*Shared Applications*.**

La ventaja de ésto es que **facilita la colaboración**. La finalidad, tanto de la API de las Aplicaciones Compartidas, junto con el resto de la API del *Toolkit de Access Grid*, es la de permitir a los desarrolladores crear aplicaciones de colaboración con más facilidad.

Las principales características que se pueden destacar de la API de las Aplicaciones Compartidas son dos:

- Las Aplicaciones Compartidas pueden compartir datos.
- Servicio de Eventos en las Aplicaciones Compartidas para enviar información entre los participantes.

Otros elementos que se pueden utilizar dentro de las Aplicaciones Compartidas son *Salas Virtuales*, obtener los participantes de una Sala, almacenamiento de ficheros, chat y servicios multimedia.

5.5.1. Introducción al desarrollo de Aplicaciones Compartidas (Shared Applications)

El conjunto de librerías que proporciona la API las Aplicaciones Compartidas es enorme, existiendo una gran cantidad de clases, tipos de datos y funciones para que el desarrollador pueda crear aplicaciones lo más fácil que sea posible. Sin embargo, la documentación generada a partir del código fuente del sistema *Access Grid* [15], está algo desorganizada y puede dificultar a la hora de que el desarrollador intente formarse en el desarrollo de *Aplicaciones Compartidas*.

Es por ello que, en este capítulo, no se pretende explicar toda la API que *Access Grid* ofrece (haría falta un gran número de volúmenes para documentar todas sus funcionalidades), sino que se intentará dar unas nociones para que el desarrollador se introduzca en el mundo del desarrollo de Aplicaciones Compartidas para *Access Grid* y, una vez introducido, por cuenta propia, se forme más profundamente en el desarrollo para este sistema, hasta que el propio desarrollador considere adecuado.

Gracias a la documentación existente sobre el desarrollo de Aplicaciones Compartidas [16], además de otros documentos ([Pms04], [Sus06]), se obtuvo información suficiente para desarrollar Aplicaciones Compartidas.

Prerrequisitos

Antes de introducirnos en cómo desarrollar Aplicaciones Compartidas, hay que asegurarse de:

- Ha instalado correctamente el *Toolkit de Access Grid 3* (ver el **5.2 - Cómo crearnos nuestro marco de trabajo**).
- Puede ejecutar una aplicación de la API (por ejemplo, *VenueClient*
- Puede conectarse a una Sala desde el Cliente de Sala (*VenueClient*)

A continuación, para facilitar un poco más la comprensión de la API del *Toolkit de Access Grid*, se mostrarán una serie de ejemplos sencillos donde se usan las clases y los métodos más usuales de *Access Grid*.

Ejemplo 1 - Una Aplicación Compartida sencilla

El primer ejemplo es bastante sencillo. Se debe crear una Aplicación Compartida, **"unirla"** y cerrar la Aplicación Compartida. Se debe entender con **Unir** una Aplicación, a la acción de **asociar la Aplicación Compartida con un Perfil de usuario**, ya que **toda Aplicación Compartida es creada por un usuario**.

El código de ejemplo es el siguiente, posteriormente, se explicará detalladamente:

```
1 # -*- coding: utf-8 -*-
2 ##########################################################################
3 #        INTRODUCCION AL DESARROLLO DE APLICACIONES COMPARTIDAS          #
4 #                          PARA ACCESS GRID                              #
5 # Ejemplo 1 - Unir una Aplicacion Compartida                             #
6 ##########################################################################
7
```

[15] http://www.accessgrid.org/files/developer/api/3.1/index.html
[16] [Vis05]

```
import agversion
agversion.select(3)

import os, sys
from AccessGrid.Venue import VenueIW
from AccessGrid.ClientProfile import ClientProfile
from AccessGrid.SharedAppClient import SharedAppClient
from AccessGrid.Platform.Config import UserConfig
from AccessGrid import Toolkit

#
# Un ejemplo simple de como unir una Aplicacion Compartida
#

# Primero hay que inicializar el toolkit de Aplicaciones Compartidas
#   de Access Grid. No siempre es necesario, pero le permite utilizar
#   varias cosas como certificados, loggin, etc.

print "- Inicializando el Toolkit de AG."
app = Toolkit.CmdlineApplication.instance()
app.Initialize("BasicApp")


def SetupTest(venueUrl):
    # La direccioin URL de la Sala donde este conectado
    venueProxy = VenueIW(venueUrl)

    # --- Crea una nueva Aplicacion Compartida en la Sala para pruebas
    appDescription = venueProxy.CreateApplication(
            "test app name",
            "test app description",
            "test app mimeType" )
    appUrl = str(appDescription.uri)
    # ---
    return venueProxy, appUrl, appDescription

def GetClientProfile():
    clientProfileFile = os.path.join(UserConfig.instance().GetConfigDir(),
                                     "profile")
    clientProfile = ClientProfile(clientProfileFile)
    return clientProfile

# Lee una direccion URL de una Sala desde la linea de comandos si es
   necesario
if len(sys.argv) > 1:
    venueUrl = sys.argv[1]
    print "- Probando en la Sala desde la linea de comandos:", venueUrl
else:
    venueUrl = "https://localhost:8000/Venues/default"
    print "- Probando en la Sala local por defecto. Introduzca una
        direccion URL de una Sala para especificar una Sala diferente."

print "- Creando una Aplicacion Compartida de prueba en la Sala."
venueProxy, appUrl, appDescription = SetupTest(venueUrl)

sharedAppClient = SharedAppClient("BasicApp")
```

```
62 sharedAppClient.InitLogging()
63
64 print "- Uniendo la Aplicacion Compartida."
65 sharedAppClient.Join(appUrl, GetClientProfile() )
66 print "- Union Finalizada"
67
68 print "- Apagando la Aplicacion Compartida."
69 # Desconecta y "apaga" la Aplicacion Compartida antes de salir (no se
      requiere pero puede ser util).
70 sharedAppClient.Shutdown()
71
72 print "- Eliminando la Aplicacion Compartida de la Sala."
73 venueProxy.DestroyApplication(appDescription.id)
74
75 print "- Finalizado."
```

Antes de hacer nada, como se ha comentado varias veces, hay que indicar a *Python* qué versión de la API *Access Grid* se desea usar, para localizar las librerías. Ésto se hacía con:

```
1     import agversion
2     agversion.select(3)
```

Luego, cualquier aplicación que vaya a utilizar la API del *Toolkit de Access Grid* debe inicializar primero el Toolkit. No se requiere siempre, pero puede ser útil ya que ésto nos permite utilizar cualquier elemento disponible en la API y que pueda ser de mucha ayuda, como certificados digitales, obtener el perfil del cliente, etc.

```
1     from AccessGrid import Toolkit
2
3     (...)
4
5     app = Toolkit.CmdlineApplication.instance()
6     app.Initialize("BasicApp")
```

Ésto hará que el *Toolkit de las Aplicaciones Compartidas de Access Grid* se inicialice y, además, todo error y toda información de depuración que se vaya generando se almacenará en el fichero llamado **'BasicApp.log'**.

`SharedAppClient` es una clase que simplifica una serie de tareas comunes que se realiza en todas las Aplicaciones Compartidas. Se crea así:

```
1     sharedAppClient = SharedAppClient("BasicApp")
2     sharedAppClient.InitLogging()
```

Para **Unir la Aplicación Compartida** creada, con el usuario que la ha creado se realiza con la siguiente función:

```
1     sharedAppClient.Join(appUrl, GetClientProfile() )
```

donde `GetClientProfile()` es una función que, a partir del fichero de configuración del usuario, se obtiene el perfil de dicho usuario.

Para finalizar, siempre que se desee destruir la aplicación, previamente hay que ”apagarla”. Ésto se consigue con la función `Shutdown()`, por último, se cerraría la aplicación.

```
    sharedAppClient.Shutdown()
```

Si se desea, puede eliminar la Aplicación Compartida creada, aunque se puede hacer desde la propia interfaz del Cliente de Sala:

```
    venueProxy.DestroyApplication(appDescription.id)
```

Para probar este ejemplo, primero, hay que **ejecutar un Cliente de Sala**, conectarse a una Sala y, por último, **adecuar este ejemplo**, introduciendo la dirección de Sala a la que se va a acceder, bien como un argumento desde una terminal o línea de comandos, o bien, desde el propio fichero fuente.

Ejemplo 2 - Obtener información general de una Aplicación Compartida

El segundo ejemplo se pide que se cree una Aplicación Compartida y se muestre, en una terminal o una línea de comandos, información relacionada de dicha Aplicación. Ésto se consigue gracias al **Estado de la Aplicación Compartida**, utilizando la función `GetApplicationState()`:

```
# -*- coding: utf-8 -*-
#############################################################################
#        INTRODUCCION AL DESARROLLO DE APLICACIONES COMPARTIDAS             #
#                          PARA ACCESS GRID                                 #
# Ejemplo 2 - Obtener informacion de una Aplicacion Compartida              #
#############################################################################

import agversion
agversion.select(3)

import os, sys
from AccessGrid.Venue import VenueIW
from AccessGrid.ClientProfile import ClientProfile
from AccessGrid.SharedAppClient import SharedAppClient
from AccessGrid.Platform.Config import UserConfig
from AccessGrid import Toolkit

#
# Obtener informacion de una Aplicacion Compartida
#

print "- Inicializando el Toolkit de AG."
app = Toolkit.CmdlineApplication.instance()
app.Initialize("GetSharedAppState")

def SetupTest(venueUrl):
    venueProxy = VenueIW(venueUrl)

    # --- Crea una nueva Aplicacion Compartida en la Sala para pruebas
    appDescription = venueProxy.CreateApplication(
            "test app name",
            "test app description",
            "test app mimeType" )
    appUrl = str(appDescription.uri)
```

```
    # ---
    return venueProxy, appUrl, appDescription

def GetClientProfile():
    clientProfileFile = os.path.join(UserConfig.instance().GetConfigDir(),
                                     "profile")
    clientProfile = ClientProfile(clientProfileFile)
    return clientProfile

# ---
# Lee una direccion URL de una Sala desde la linea de comandos si es
   necesario
if len(sys.argv) > 1:
    venueUrl = sys.argv[1]
    print "- Probando en la Sala desde la linea de comandos:", venueUrl
else:
    venueUrl = "https://localhost:8000/Venues/default"
    print "- Probando en la Sala local por defecto. Introduzca una
       direccion URL de una Sala para especificar una Sala diferente."

print "- Creando una Aplicacion Compartida de prueba en la Sala."
venueProxy, appUrl, appDescription = SetupTest(venueUrl)

sharedAppClient = SharedAppClient("GetSharedAppState")
sharedAppClient.InitLogging()

print "- Uniendo la Aplicacion Compartida."
sharedAppClient.Join(appUrl, GetClientProfile() )
print "- Union Finalizada"

print "- Obteniendo el Estado de la Aplicacion Compartida:"
state = sharedAppClient.GetApplicationState()
print "  Estado de la Aplicacion Compartida:"
print '    nombre:', state.name
print '    desc:', state.description
print '    id:', state.id
print '    mimeType:', state.mimeType
print '    uri:', state.uri
print '    data:', state.data

print "- Apagando la Aplicacion Compartida."
# Desconecta y "apaga" la Aplicacion Compartida antes de salir (no se
   requiere pero puede ser util).
sharedAppClient.Shutdown()

print "- Eliminando la Aplicacion Compartida de la Sala."
venueProxy.DestroyApplication(appDescription.id)

print "- Finalizado."
```

Ejemplo 3 - Compartir Datos en una Aplicación Compartida

En el siguiente ejemplo se muestra cómo las Aplicaciones Compartidas en *Access Grid* pueden compartir datos. **`SharedAppData`** es un lugar compartido por todos los participantes donde las Apli-

caciones Compartidas almacenan información. Normalmente se suele utilizar para **almacenar estados** que el resto de participantes pueden necesitar saber. Por ejemplo, si tenemos un navegador web compartido, la dirección URL actual que se está visitando es almacenado en dicho lugar. Así, el resto de participantes o, incluso si se añaden nuevos participantes más tarde, obtienen dicha URL y acceden a la misma página web.

```
# -*- coding: utf-8 -*-
#############################################################################
#        INTRODUCCION AL DESARROLLO DE APLICACIONES COMPARTIDAS              #
#                          PARA ACCESS GRID                                 #
# Ejemplo 3 - Compartir Datos en una Aplicacion Compartida                  #
#############################################################################

import agversion
agversion.select(3)

import os, sys
from AccessGrid.Venue import VenueIW
from AccessGrid.ClientProfile import ClientProfile
from AccessGrid.SharedAppClient import SharedAppClient
from AccessGrid.Platform.Config import UserConfig
from AccessGrid import Toolkit

#
# Establecer y obtener un dato compartido en una Aplicacion Compartida
#

print "- Inicializando el Toolkit de AG."
app = Toolkit.CmdlineApplication.instance()
app.Initialize("SharedAppVenueDataExample")

# Read a venueUrl from the cmd-line if necessary.
if len(sys.argv) > 1:
    venueUrl = sys.argv[1]
    print "- Probando en la Sala desde la linea de comandos:", venueUrl
else:
    venueUrl = "https://localhost:8000/Venues/default"
    print "- Probando en la Sala local por defecto. Introduzca una
        direccion URL de una Sala para especificar una Sala diferente."
venueProxy = VenueIW(venueUrl)

print "- Creando una Aplicacion Compartida de prueba en la Sala."
appDescription = venueProxy.CreateApplication(
            "test app name",
            "test app description",
            "test app mimeType" )
appUrl = appDescription.uri

print "- Uniendo la Aplicacion Compartida."
sharedAppClient = SharedAppClient("testApp")
sharedAppClient.InitLogging()

# Obtenemos el perfil del cliente
clientProfileFile = os.path.join(UserConfig.instance().GetConfigDir(),
                                 "profile")
```

```
clientProfile = ClientProfile(clientProfileFile)
sharedAppClient.Join(appUrl, clientProfile)

print "-- Estableciendo dato compartido:"
testKey = "test key"
testValue = "test value"
print "   ", testKey, ":", testValue
sharedAppClient.SetData( testKey, testValue )

print "-- Obteniendo llaves de los datos compartidos:"
keys = sharedAppClient.GetDataKeys()
for k in keys:
    print "   ", k

print "-- Obteniendo datos compartidos:"
retVal = sharedAppClient.GetData( keys[0] )
print "   ", keys[0], ":", retVal
if not retVal == testValue:
    print "ERROR: El valor obtenido es diferente al valor que se establecio
        "

print "- Apagando la Aplicacion Compartida."
# Desconecta y "apaga" la Aplicacion Compartida antes de salir (no se
    requiere pero puede ser util).
sharedAppClient.Shutdown()

print "- Eliminando la Aplicacion Compartida de la Sala."
venueProxy.DestroyApplication(appDescription.id)

print "- Finalizado."
```

Observe cómo se establecen datos para compartir y cómo se obtienen datos compartidos. Para compartir un dato:

```
    testKey = "test key"
    testValue = "test value"

    sharedAppClient.SetData( testKey, testValue )
```

Para obtener un dato compartido:

```
    value = sharedAppClient.SetData( "test key" )
```

Por último, para obtener todos los datos compartidos de una Aplicación Compartida:

```
    keys = sharedAppClient.GetDataKeys()
    print keys
```

Se recomienda modificar el código fuente para asignar nuevos datos compartidos.

Ejemplo 4 - Eventos Compartidos

En el próximo ejemplo, se mostrará cómo utilizar los **Eventos Compartidos** en las Aplicaciones Compartidas. Como se ha comentado en el ejemplo 1, la clase `sharedAppClients` representa, de

manera lógica, una Aplicación Compartida. Por cada cliente, se instancia un objeto de dicha clase, así, pues, los Eventos Compartidos permiten **comunicarse entre ellos**.

```
# -*- coding: utf-8 -*-
#############################################################################
#        INTRODUCCION AL DESARROLLO DE APLICACIONES COMPARTIDAS              #
#                          PARA ACCESS GRID                                 #
# Ejemplo 4 - Eventos Compartidos                                           #
#############################################################################

import agversion
agversion.select(3)

import os, traceback, sys
from AccessGrid.Venue import VenueIW
from AccessGrid.ClientProfile import ClientProfile
from AccessGrid.SharedAppClient import SharedAppClient
from AccessGrid.Platform.Config import UserConfig
from AccessGrid import Toolkit

try:
    from twisted.internet import _threadedselect as threadedselectreactor
except:
    from twisted.internet import threadedselectreactor
try:
    threadedselectreactor.install()
except:
    pass

#
# Enviar y recibir un Evento de una Aplicacion Compartida
#

print "- Inicializando el Toolkit de AG."
app = Toolkit.CmdlineApplication.instance()
app.Initialize("AppEvents")

def SetupTest(venueUrl):
    venueProxy = VenueIW(venueUrl)

    # --- Crea una nueva Aplicacion Compartida en la Sala para pruebas
    appDescription = venueProxy.CreateApplication(
            "test app name",
            "test app description",
            "test app mimeType" )
    appUrl = str(appDescription.uri)
    # ---
    return venueProxy, appUrl, appDescription

def GetClientProfile():
    clientProfileFile = os.path.join(UserConfig.instance().GetConfigDir(),
                                     "profile")
    clientProfile = ClientProfile(clientProfileFile)
    return clientProfile
```

```
52
53
54  # Lee una direccion URL de una Sala desde la linea de comandos si es
        necesario
55  if len(sys.argv) > 1:
56      venueUrl = sys.argv[1]
57      print "- Probando en la Sala desde la linea de comandos:", venueUrl
58  else:
59      venueUrl = "https://localhost:8000/Venues/default"
60      print "- Probando en la Sala local por defecto. Introduzca una
            direccion URL de una Sala para especificar una Sala diferente."
61
62  print "- Creando una Aplicacion Compartida de prueba en la Sala."
63  venueProxy, appUrl, appDescription = SetupTest(venueUrl)
64
65  sharedAppClient = SharedAppClient("AppEvents")
66  sharedAppClient.InitLogging()
67
68  print "- Uniendo la Aplicacion Compartida."
69  sharedAppClient.Join(appUrl, GetClientProfile() )
70  print "- Union Finalizada"
71
72  def shutdown():
73      print "- Apagando la Aplicacion Compartida."
74      reactor.stop()
75
76      # Desconecta y "apaga" la Aplicacion Compartida antes de salir (no se
            requiere pero puede ser util).
77      sharedAppClient.Shutdown()
78
79      print "- Eliminando la Aplicacion Compartida de la Sala."
80      venueProxy.DestroyApplication(appDescription.id)
81
82      print "- Finished."
83
84  def event1Callback(eventDesc):
85      try:
86          print "- Evento Recibido."
87          print "    Tipo: %s," % eventDesc.GetEventType(), "Dato:",
                eventDesc.GetData()
88          shutdown()
89      except:
90          traceback.print_exc()
91
92  sharedAppClient.RegisterEventCallback("event1", event1Callback )
93
94  def sendEvent():
95      print "- Enviando Evento."
96      print "    Tipo: event1, Dato: somedata"
97      sharedAppClient.SendEvent("event1", "somedata")
98
99  # Llama una funcion una vez que el evento del cliente ha sido conectado
100 sharedAppClient.eventClient.RegisterMadeConnectionCallback(sendEvent)
101
102 from twisted.internet import reactor
103 reactor.run()
```

La Aplicación Compartida del ejemplo envía un evento, pero, antes de nada, **hay que registrar ese evento**.

```
    def event1Callback():
        print "- Received event."
        sharedAppClient.RegisterEventCallback("event1", event1Callback )
```

Después, se enviará el evento, para ello se hace lo siguiente:

```
    def sendEvent():
        sharedAppClient.SendEvent("event1", "somedata")
```

Observe que el sistema *Access Grid* **utiliza conexiones asíncronas** utilizando, para ello, el *Toolkit de Twisted*. Como ya sabrá, una conexión asíncrona implica que el código no se quedará bloqueado esperando una llamada para finalizar y continuar procesando las siguientes instrucciones que vienen a continuación. Normalmente solicitamos a la aplicación que nos notifique cuando una llamada de red se ha completado. En nuestro caso, se solicitará tal confirmación una vez que la conexión se ha realizado, de la siguiente manera:

```
    sharedAppClient.eventClient.RegisterMadeConnectionCallback(sendEvent)
```

Ejemplo 5 - Almacenamiento de Ficheros en una Sala

En este ejemplo se mostrará cómo una Aplicación Compartida accede a los ficheros compartidos de una Sala, bien para subir un fichero, o bien para descargárselo o incluso borrarlo.

```
# -*- coding: utf-8 -*-
#############################################################################
#        INTRODUCCION AL DESARROLLO DE APLICACIONES COMPARTIDAS             #
#                          PARA ACCESS GRID                                 #
# Ejemplo 5 - Almacenamiento de Ficheros en una Sala                        #
#############################################################################

import agversion
agversion.select(3)

import os, sys
from AccessGrid.Venue import VenueIW
from AccessGrid.ClientProfile import ClientProfile
from AccessGrid.SharedAppClient import SharedAppClient
from AccessGrid.Platform.Config import Config, UserConfig
from AccessGrid.DataStoreClient import GetVenueDataStore
from AccessGrid.Platform.Config import SystemConfig
from AccessGrid.interfaces.VenueClient_client import VenueClientIW
from AccessGrid.GUID import GUID
from AccessGrid import Toolkit

from AccessGrid.Toolkit import CmdlineApplication

print "- Inicializando el Toolkit de AG."
Toolkit.CmdlineApplication()
```

```
26  app = Toolkit.CmdlineApplication.instance()
27  app.Initialize("SharedAppVenueDataExample")
28
29  #
30  # Accediendo a los ficheros compartidos de una Sala y subir o bajar un
        fichero
31  #
32
33  if len(sys.argv) < 3:
34      print "  <venueUrl> <connectionID> no especificado en la linea de
            comandos."
35      from AccessGrid.VenueClient import GetVenueClientUrls
36      venueClientUrls = GetVenueClientUrls()
37      print "  Se le solicitara que introduzca un connecctionId y una
            direccion URL de una Sala"
38      if len(venueClientUrls) > 0:
39          venueClient = VenueClientIW(venueClientUrls[0])
40          connectionId = venueClient.GetClientProfile().connectionId
41          print "  ConnectionId del VenueClient : ", connectionId
42          venueURL = str(venueClient.GetVenueURL())
43          print "  URL del VenueClient : ", venueURL
44      else:
45          print "Especifique un connectionId en la linea de comandos o
                conecte un Cliente en una sala."
46          sys.exit()
47  else:
48      connectionId = sys.argv[1]
49
50
51  def GetClientProfile():
52      clientProfileFile = os.path.join(UserConfig.instance().GetConfigDir(),
53                                       "profile")
54      clientProfile = ClientProfile(clientProfileFile)
55      return clientProfile
56
57
58  # La direccioin URL de la Sala donde este conectado
59  venueUrl = "https://localhost:8000/Venues/default"
60  venueProxy = VenueIW(venueUrl)
61
62  print "- Creando una Aplicacion Compartida de prueba en la Sala."
63  appDescription = venueProxy.CreateApplication(
64              "test app name",
65              "test app description",
66              "test app mimeType" )
67  appUrl = str(appDescription.uri)
68
69  sharedAppClient = SharedAppClient("SharedAppVenueData")
70  sharedAppClient.InitLogging()
71
72  print "- Uniendo la Aplicacion Compartida."
73  sharedAppClient.Join(appUrl, GetClientProfile() )
74  print "- Union Finalizada"
75
76  dataStoreClient = GetVenueDataStore(venueUrl, connectionId)
77
```

```
78  print "- Consultando los ficheros almacenados en la Sala: "
79  filenames = dataStoreClient.QueryMatchingFiles("*")
80  print "    Ficheros de la Sala:", filenames
81
82  # Crea un fichero a subir
83  hostname = SystemConfig.instance().GetHostname()
84  filename = "tmp_%s.txt" % hostname
85  f = open(filename, "w").write("Esto es un fichero de prueba.\n")
86  print "- Subiendo: ", filename
87  dataStoreClient.Upload(filename)
88
89  print "- Actualizando la lista de ficheros almacenados."
90  dataStoreClient.LoadData()
91
92  print "- Consultando de nuevo los ficheros almacenados en la Sala (."
93  filenames = dataStoreClient.QueryMatchingFiles("*")
94  print "    Ficheros de la Sala:", filenames
95
96  print "- Dato del Fichero:\n    ", dataStoreClient.GetFileData(filename)
97
98  print "- Descargando: ", filename
99  downloadFilename = "descargado_" + filename
100 dataStoreClient.Download(filename, downloadFilename)
101
102 print "- Eliminando fichero de la Sala."
103 dataStoreClient.RemoveFile(filename)
104
105 print "- Eliminando la Aplicacion Compartida de la Sala."
106 venueProxy.DestroyApplication(appDescription.id)
107
108 print "- Finalizado."
```

Para acceder al almacén de ficheros de una Sala, la Aplicación Compartida necesita la dirección URL de dicha Sala, que proporciona la localización de la Sala y, además, un **`connectionId`** para demostrar que el usuario tiene permiso para acceder a la Sala. Un `connectionId` se recibe cuando se establece una conexión con una Sala, normalmente por un Cliente de Sala (*VenueClient*). El `connectionId` y la dirección URL de la Sala se les pasa a la Aplicación Compartida.

Otra forma de conseguir el `connectionId` es a través de un Cliente de Sala conectado en la máquina local.

```
1     venueClient = VenueClientIW(venueClientUrls[0])
2     connectionId = venueClient.GetClientProfile().connectionId
```

Una vez obtenidas la dirección de Sala y el `connectionId`, podemos conseguir el almacén de ficheros de la Sala:

```
1     dataStoreClient = GetVenueDataStore(venueUrl, connectionId)
```

Una vez obtenido el almacén de ficheros, podemos hacer las siguientes operaciones:

- **Consultar los ficheros existentes:**

```
1     filenames = dataStoreClient.QueryMatchingFiles("*")
```

- **Descargar un fichero existente:**

```
1       dataStoreClient.Download(filename, localDownloadFilename
```

- **Subir un fichero:**

```
1       dataStoreClient.Upload(filename)
2       # Recarga la lista de ficheros
3       dataStore.LoadData()
```

- **Borrar un fichero:**

```
1       dataStoreClient.RemoveFile(filename)
```

Ejemplo 6 - Aplicaciones Compartidas con Interfaz Gráfica de Usuario

Como se ha comentado, *Access Grid*, además de estar desarrollado en *Python*, sus interfaces gráficas están desarrolladas bajo *wxPython* [17].

No obstante, aunque el núcleo del sistema *Access Grid* esté desarrollado bajo *wxPython*, **existe un método de poder utilizar otro tipo de interfaz**, como, por ejemplo *Qt4*. En próximos capítulos se explicará este método.

El siguiente ejemplo muestra una simple Aplicación Compartida con interfaz gráfica diseñada bajo *wxPython*.

```
1  # -*- coding: utf-8 -*-
2  ##########################################################################
3  #        INTRODUCCION AL DESARROLLO DE APLICACIONES COMPARTIDAS          #
4  #                          PARA ACCESS GRID                              #
5  # Ejemplo 6 - Aplicacion Compartida con Interfaz Grafica de Usuario      #
6  ##########################################################################
7
8  import agversion
9  agversion.select(3)
10
11 import os, sys, traceback
12 from AccessGrid.Venue import VenueIW
13 from AccessGrid.ClientProfile import ClientProfile
14 from AccessGrid.SharedAppClient import SharedAppClient
15 from AccessGrid.Platform.Config import Config, UserConfig
16 from AccessGrid.DataStoreClient import GetVenueDataStore
17 from AccessGrid.Platform.Config import SystemConfig
18 from AccessGrid.interfaces.VenueClient_client import VenueClientIW
19 from AccessGrid.GUID import GUID
20 from AccessGrid import Toolkit
21 from AccessGrid import icons
22 from wxPython.wx import *
23
```

[17] http://www.wxpython.org/docs/api/

```
from AccessGrid.Toolkit import WXGUIApplication
try:
    from twisted.internet import _threadedselect as threadedselectreactor
except:
    from twisted.internet import threadedselectreactor
try:
    threadedselectreactor.install()
except:
    pass
from twisted.internet import reactor

class SharedAppExample(wxApp):
    def __init__( self, appUrl):
        wxApp.__init__(self, False)
        self.reactorFinished = []

        reactor.interleave(wxCallAfter)

        name = sys.argv[0]
        self.sharedAppClient = SharedAppClient(name)
        self.log = self.sharedAppClient.InitLogging()

        # Obtenemos el perfil del cliente
        try:
            clientProfileFile = os.path.join(UserConfig.instance().
                GetConfigDir(), "profile")
            clientProfile = ClientProfile(clientProfileFile)
        except:
            self.log.info("SharedAppClient.Connect: No ha sido posible
                obtener el perfil del cliente, se establecera clientProfile
                = None")
            clientProfile = None

        print "- Uniendo la Aplicacion Compartida."
        self.sharedAppClient.Join(appUrl, clientProfile)
        print "- Union Finalizada"

        # Indicamos que la Aplicacion Compartida tendra un evento y se
            llamara a la funcion
                # self.madeConnectionCallback cuando dicho evento suceda
        self.sharedAppClient.eventClient.RegisterMadeConnectionCallback(
            self.madeConnectionCallback)

        self.frame = wxFrame(None, -1, "Shared App Example")

        self.hsizer = wxStaticBoxSizer(wxStaticBox(self.frame, -1, "Botones
            "), wxHORIZONTAL)
        self.button1Id = wxNewId()
        self.button1 = wxButton(self.frame, self.button1Id,
                        "Boton 1", wxDefaultPosition, wxDefaultSize)
        self.quitButtonId = wxNewId()
        self.quitButton = wxButton(self.frame, self.quitButtonId,
                        "Salir", wxDefaultPosition, wxDefaultSize)

        self.mainSizer=wxBoxSizer(wxVERTICAL);
        self.mainSizer.Add(self.hsizer, 0, wxALIGN_CENTER | wxGROW | wxALL,
```

```
            5)
74          self.hsizer.Add(self.button1, 0, wxALIGN_CENTER | wxGROW | wxALL,
                5)
75          self.hsizer.Add(self.quitButton, 0, wxALIGN_CENTER | wxGROW | wxALL
                , 5)
76
77          EVT_BUTTON(self.frame, -1, self.OnButton)
78          EVT_BUTTON(self.frame, self.quitButtonId, self.Quit)
79          self.frame.SetSizer(self.mainSizer);
80          self.frame.SetAutoLayout(true);
81
82          self.frame.SetIcon(icons.getAGIconIcon())
83          self.frame.Show(1)
84          self.SetTopWindow(self.frame)
85
86      def OnButton(self, event):
87          if event.GetId() == self.button1Id:
88              print "  Boton 1 pulsado"
89          else:
90              print "  Boton desconocido cuya id es:", event.GetId()
91
92      def madeConnectionCallback(self):
93          try:
94              self.log.debug("MadeConnection")
95              print "  Conexion con el evento del cliente realizada. (Pulsar
                    Salir para continuar.)"
96          except:
97              traceback.print_exc()
98              self.log.exception()
99
100     def Quit(self, event=None):
101         print "  Boton Salir pulsado."
102         self.frame.Close()
103
104     def ReactorFinished(self):
105         self.reactorFinished.append(True)
106
107     def OnExit(self):
108         print "- Apagando la Aplicacion Compartida."
109         reactor.stop()
110
111         try:
112             self.sharedAppClient.Shutdown()
113         except:
114             traceback.print_exc()
115
116         # Dejamos finalizar el reactor twisted
117         while len(self.reactorFinished) == 0:
118             reactor.iterate()
119
120 if __name__ == "__main__":
121
122     print "- Inicializando el Toolkit de AG."
123     app = Toolkit.WXGUIApplication.instance()
124     app.Initialize(sys.argv[0])
125
```

```
126     # Read a venueUrl from the cmd-line if necessary.
127     if len(sys.argv) > 1:
128         venueUrl = sys.argv[1]
129         print "- Probando en la Sala desde la linea de comandos:", venueUrl
130     else:
131         venueUrl = "https://localhost:8000/Venues/default"
132         print "- Probando en la Sala local por defecto. Introduzca una
              direccion URL de una Sala para especificar una Sala diferente."
133
134     venue = VenueIW(venueUrl)
135
136     print "- Creando una Aplicacion Compartida de prueba en la Sala."
137     appDescription = venue.CreateApplication(
138             "test app name",
139             "test app description",
140             "test app mimeType" )
141     appUrl = str(appDescription.uri)
142
143
144     wxInitAllImageHandlers()
145     app = SharedAppExample(appUrl)
146     reactor.addSystemEventTrigger('after', 'shutdown', app.ReactorFinished)
147     app.MainLoop()
148
149     print "- Eliminando la Aplicacion Compartida de la Sala."
150     # Remove the test shared app from the venue.
151     try:
152         venue.DestroyApplication(appDescription.id)
153         print "- Aplicacion Compartida eliminada de la Sala."
154     except:
155         traceback.print_exc()
156
157     print "- Finalizado."
158
159     os._exit(0)
```

De forma resumida, se puede indicar los siguientes pasos a seguir:

- Se crea una clase heredando la clase `wxApp`, junto al propio diseño de la aplicación. En nuestro caso, se trata de **una ventana con dos botones**:

```
1       class SharedAppExample(wxApp):
2           def __init__( self, appUrl):
3               wxApp.__init__(self, False)
4               reactor.interleave(wxCallAfter)
5               ...
6               self.sharedAppClient = SharedAppClient("Shared App Example
                    ")
7               self.log = self.sharedAppClient.InitLogging()
8               self.sharedAppClient.Join(appUrl, clientProfile)
9
10              #Registramos un evento de la aplicacion y lo asociamos con
                     la funcion
11              # self.madeConnectionCallback
12              self.sharedAppClient.eventClient.
                    RegisterMadeConnectionCallback(
```

```
                    self.madeConnectionCallback)

            self.frame = wxFrame(None, -1, "Shared App Example")
            ... # creacion de la interfaz
            EVT_BUTTON(self.frame, -1, self.OnButton)
            EVT_BUTTON(self.frame, self.quitButtonId, self.Quit)
            ...
```

- Se definen métodos para cerrar la aplicación cuando se sale de ella:

```
    def Quit(self, event=None):
        self.frame.Close()

    def OnExit(self):
        reactor.stop()
        try:
            self.sharedAppClient.Shutdown()
        except:
            traceback.print_exc()

        # Dejamos finalizar el reactor twisted
        while len(self.reactorFinished) == 0:
            reactor.iterate()
```

- Una vez definida la clase, se procede a inicializarla:

```
    app = Toolkit.WXGUIApplication.instance()
    app.Initialize("appName")
```

- Por último, comienza la Aplicación Compartida:

```
    wxInitAllImageHandlers()
    app = SharedAppExample(appUrl)
    reactor.addSystemEventTrigger('after', 'shutdown', app.
       ReactorFinished)
    app.MainLoop()
```

5.5.2. Empaquetamiento y Distribución de una Aplicación Compartida

Una vez desarrollada una Aplicación Compartida, como es habitual, hay que empaquetarla y distribuirla para que cualquiera pueda instalarla lo más fácilmente posible. Normalmente se utilizan creadores de instaladores para tal fin, donde, mediante unos *scripts* se generan las ventanas que conforman dicho instalador y las acciones que realiza en cada una de ellas.

En este caso, podemos utilizar los instaladores, sin embargo, la API del *Toolkit de Access Grid*, proporciona una herramienta para **empaquetar fácilmente nuestras aplicaciones** y, luego, un sistema instalado de *Access Grid* trae soporte para detectar dicho empaquetamiento y **automáticamente desempaquetará el software y lo integrará en *Access Grid***.

Para ello hay que realizar lo siguiente:

Generar un fichero *app*

Antes de nada, hay que **tener todos los ficheros que conforman el software en una carpeta**, además, por limitaciones del empaquetador de *Access Grid*, estos ficheros **no pueden estar en subcarpetas**. Es una limitación que se encontró cuando se desarrollaron nuestras aplicaciones, por ejemplo, existía una carpeta "images" donde se almacenaban los iconos de las ventanas del software. Sin embargo, a la hora de empaquetar y distribuir el software, el empaquetador de *Access Grid* no detectaba dichas carpetas.

Una vez se tenga todo el código fuente organizado en una carpeta, en dicha carpeta, se crea un fichero con extensión *.app*, **el nombre del fichero debe coincidir con el nombre de la aplicación**. Por ejemplo, si la aplicación se llama *SharedApp1*, el fichero deberá llamarse *SharedApp1.app*. El contenido de este fichero tiene la siguiente estructura:

```
[application]
name = Nombre de la aplicacion
mimetype = application/x-ag-nombre-de-la-aplicacion
extension = nombredelaaplicacion
files = Conjunto de ficheros que forman la aplicacion

[commands]
Open = Forma de ejecutar la aplicacion, normalmente, siguiendo la
    sintaxis que suelen hacer
        el resto de aplicaciones, suele ser:
        %(python)s FicheroPrincipal.py -a %(appUrl)s -v %(venueUrl)s
```

Una vez completado los campos, se guardan los cambios.

Crear un fichero *agpkg3*

Para finalizar de empaquetar nuestra aplicación, hay que generar un fichero con extensión *agpkg3* (*Access Grid Package 3*). En algunos sistemas tiene la extensión *agpkg*, pero el método a seguir es prácticamente el mismo.

Simplemente hay que seleccionar todos los ficheros que conforman la Aplicación Compartida, más el fichero con extensión *.app* y **comprimirlos en un fichero *zip***. Una vez generado el fichero comprimido con extensión *.zip*, **se cambia la extensión a *agpkg3* o a *agpkg***, dependiendo del sistema operativo.

5.5.3. Instalación de una Aplicación Compartida

Una vez se tenga empaquetada la Aplicación Compartida que se desea distribuir, a través del sistema de empaquetamiento de *Access Grid*, instalarla también es muy fácil.

Sistemas Linux

Simplemente, basta con introducir el siguiente comando en una terminal:

```
sudo agpm3 -s -p AplicacionCompartida.agpkg3
```

Automáticamente, se descomprimirá el fichero en la carpeta apropiada del sistema *Access Grid* y, por último, integrará la Aplicación Compartida con *Access Grid*.

Sistemas Windows

Instalar una Aplicación Compartida en los sistemas Windows es mucho más fácil aún, ya que, cuando se instala el sistema *Access Grid*, el sistema asocia automáticamente los ficheros con extensión *.agpkg3* con su herramienta de instalación, que es la aplicación *agpm3.py* (*Access Grid Package Manager*). Así que, para instalar una Aplicación Compartida, **basta con hacer doble click en el fichero con extensión *.agpkg3***.

Automáticamente, se descomprimirá el fichero en la carpeta apropiada del sistema *Access Grid* y, por último, integrará la Aplicación Compartida con *Access Grid*.

5.6. Shared Hello World

Para entender mucho mejor cómo desarrollar Aplicaciones Compartidas, antes de realizar ninguna aplicación, se optó por desarrollar primero una aplicación de ejemplo. Igual que todo desarrollador, cuando conoce un nuevo lenguaje o una nueva herramienta, lo primero que desarrolla es un *Hola Mundo*, yo, como desarrollador iniciado de Aplicaciones Compartidas para *Access Grid*, también se creó un *Hola Mundo*, concretamente se llama *Shared Hello World*.

Figura 5.15: Aplicación Compartida Shared Hello World

En un primer diseño, la aplicación, al pulsar sobre el botón *Enviar Evento*, escribirá el texto ”Hola Mundo” tanto al propio usuario, como al resto de participantes. Así, se comprobaba el correcto funcionamiento de los Eventos Compartidos.

Sin embargo, como desarrollador, interesaba ”jugar” un poco más con los Eventos Compartidos, y comprobar que realmente funcionaba en la aplicación desarrollada.

Así que, se modificó la aplicación *Shared Hello World*, de forma que, el que pulse el botón le aparecerá un texto ”Evento Enviado”, y, al resto de participantes recibirán un texto diferente: ”Evento Recibido”.

5.6.1. Paso a paso

A continuación, para que el desarrollador entienda mejor cómo se desarrollo la aplicación, se explicarán los pasos seguidos.

Diseño de la Interfaz Gráfica de Usuario

Lo primero que se desarrolló fue la Interfaz de la Aplicación Compartida. Existen herramientas que facilitan este proceso, como por ejemplo ***wxGlade***, aunque si lo desea, puede crear su interfaz directamente **desde un editor de textos**.

La interfaz de la aplicación *Shared Hello World* es la siguiente:

Figura 5.16: Interfaz de la Aplicación Compartida Shared Hello World

Para tener mejor organizado el código fuente de la Aplicación, se recomienda **separar el código fuente de la Interfaz Gráfica del resto de la aplicación**. Así, se creó el fichero `HelloWorld_GUI.py` que contiene el código fuente de la interfaz.

Desarrollo de la Aplicación Compartida

A continuación, se comienza a desarrollar la Aplicación Compartida, es aconsejable seguir el esqueleto explicado en el **ejemplo 6** de la **sección 5.5.1**.

Hay que destacar que debemos **establecer los Eventos Compartidas**. Ésto puede parecer complicado, sin embargo, siguiendo unas pautas, resulta bastante sencillo de implementar.

- **¿Qué evento se desea realizar?:** En nuestro caso, enviar un texto tanto al usuario, como al resto de participantes, con un texto diferente a cada uno.
- **Registrar el Evento Compartido:** Una vez que se sabe qué eventos se desean compartir, se registran indicando qué función habría que llamar cuando se desee enviar el Evento Compartido.

```
1       self.sharedAppClient.RegisterEventCallback("NewText",self.
            HandleBtnEvento)
```

  En nuestro caso, se llamará a la función `HandleBtnEvento` cuando enviemos el Evento Compartido "New Text".

- **¿Cuándo se desea que suceda el evento?:** En nuestro caso, queremos que el texto se envíe al pulsar el botón *Enviar Evento*. Así pues, hay que asignar al botón *Enviar Evento*, un evento de tipo *Botón Pulsado* y se le asociará una función, que será quien escriba el texto y lo envíe al resto de participantes.

```
1       # Eventos de la aplicacion
2       self.frame.Bind(wx.EVT_BUTTON, self.OnBtnEventoClick)
```

- **Implementar la función asociada al Evento Compartida:** En nuestro caso, únicamente queremos escribir un texto en un cuadro de texto de la interfaz, diferenciando quién ha pulsado el botón del resto. El texto **se enviará a través de un *Dato Compartido*** (la variable `markupText`), que viajará al resto de participantes para ser escrito en sus ventanas.

```
1       def HandleBtnEvento(self,eventdata):
2           print "Recibiendo evento al resto de participantes"
3
4           #senderId almacena quien ha pulsado el boton
5           senderId = eventdata.GetSenderId()
```

```
6           print "SenderId = ", senderId
7
8           # Si el usuario no es el que lo ha pulsado
9           if senderId != self.sharedAppClient.GetPublicId():
10              print "El usuario no es el que lo ha pulsado"
11              try:
12                  self.markupText = self.sharedAppClient.GetData("txt")
13                  self.frame.edText.SetValue(self.markupText)
14              except:
15                  self.log.exception("SharedHelloWorld: petada de texto"
                        )
16          # Si no
17          else:
18              print "El usuario es el que lo ha pulsado"
19              try:
20                  self.markupText = "Evento enviado"
21                  self.frame.edText.SetValue(self.markupText)
22              except:
23                  self.log.exception("SharedHelloWorld: petada de texto"
                        )
```

- **Para finalizar, se implementa el Evento de la Aplicación:** En nuestro caso, se implementará el evento de **Botón Pulsado**, donde se llamará al Evento Compartido implementado anteriormente.

```
1       def OnBtnEventoClick(self, event):
2           print "Boton compartido pulsado"
3           self.markupText = "Evento recibido"
4
5           #Establecemos el Dato Compartido
6           print "  A enviar: txt", ":", self.markupText
7           self.sharedAppClient.SetData("txt",self.markupText)
8
9           print "PublicId = ", self.PublicId
10
11          # Enviamos el evento a la Sala para que surta efecto en el
                resto de participantes
12          self.sharedAppClient.SendEvent("NewText", self.PublicId)
13          print "Enviando evento al resto de participantes"
```

5.6.2. Resultado Final

Una vez explicado el desarrollo de la Aplicación Compartida, el resultado final, incluyendo el fichero *SharedHelloWorld.app* para su empaquetamiento y distribución, se encuentra disponible en la siguiente dirección web: `https://forja.rediris.es/plugins/scmsvn/viewcvs.php/SharedApplications/SharedHelloWorld/?root=cusl4-aguca`.

Distribución de la Aplicación Compartida

A modo de recordatorio, a continuación se indican los pasos a seguir para empaquetar y distribuir la aplicación *Shared Hello World*.

- Crear un fichero comprimido, en formato *zip*, con los ficheros `HelloWorld_GUI.py`, `SharedHelloWorld.py` y `SharedHelloWorld.app`

- Cambiar la extensión *.zip* por la extensión *.agpkg* o *.agpkg3*, según el sistema operativo donde se vaya a instalar.

Por último, para su instalación, hay que seguir los pasos indicados en la **sección 5.5.3**.

5.7. Visor de Documentos PDF

Después de haber explicado lo mejor posible cómo desarrollar Aplicaciones Compartidas para *Access Grid*, a continuación, se mostrará una Aplicación Compartida "seria", de uso final (de hecho, **todas las Universidades Andaluzas han obtenido el software y lo están utilizando**).

Su nombre es ***Shared PDF Viewer*** y, como es de suponer, se trata de un **visor de documentos PDFs compartido**. Esta Aplicación Compartida es un visor de documentos PDFs simple (de momento, ya que puede desarrollarse futuras versiones añadiendo nuevas características), cuya finalidad es la de poder visualizar un documento y poder compartirlo con el resto de participantes.

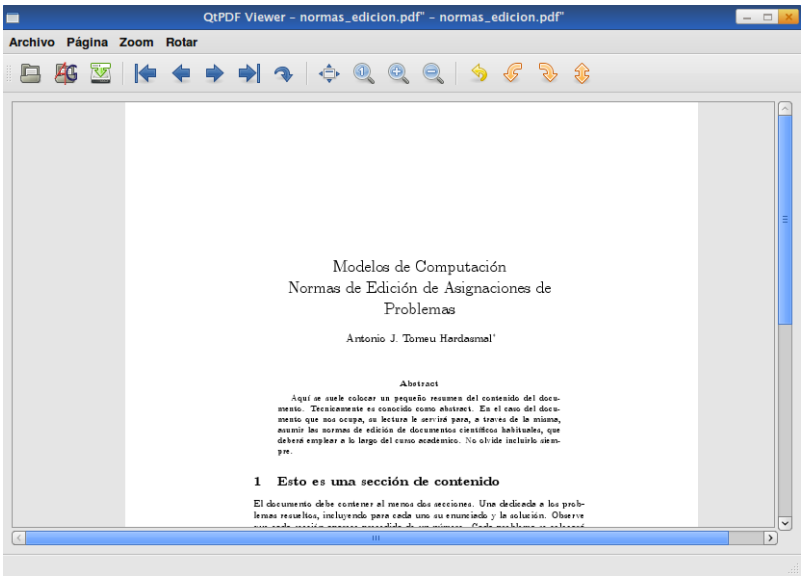


Figura 5.17: Aplicación Shared PDF Viewer

Esta Aplicación Compartida surge a partir de los objetivos primarios que se estableció al inicio del proyecto. Hay que recordar que uno de los objetivos era poder **realizar alguna aplicación que pudiera cargar presentaciones** y que el resto de participantes pudieran seguir el flujo de la presentación, visualizando el documento en sus equipos.

Sin embargo, el formato *.ppt o .pptx* (de *PowerPoint*) no es un formato abierto y tampoco existe una gran diversidad de visores de dichos formatos. Por lo que desarrollar una Aplicación Compartida para tal fin resultaba, a primera vista, algo complicado.

Así pues, se optó por desarrollar un visor de documentos en PDF, con la finalidad de que se usara para **compartir presentaciones entre los participantes**.

Esto se puede hacer de dos maneras, una de ellas es que el profesor, ponente, etc, en definitiva, aquella persona que desee desarrollar una presentación, **exportar su documento a PDF**, cosa bastante fácil hoy en día ya que, hasta el propio *Microsoft Office* puede exportar a PDF sus documentos.

La otra alternativa, aún mucho más cómoda, es que, la propia aplicación *Shared PDF Viewer*, internamente, al cargar un documento en formato *.doc ó .ppt*, realizará una conversión de dicho documento a formato *PDF* y lo podrá visualizar [18].

5.7.1. Características

Como ya se ha comentado, esta Aplicación Compartida consiste en un visor de documentos PDF compartido. ¿Qué se quiere decir con compartido?. Pues que habrá eventos en dicho visor que afecte al resto de participantes, es decir, **la Aplicación Compartida tiene varios Eventos Compartidos**. Éstos son los siguientes:

Ejecución de la Aplicación Compartida

La aplicación *Shared PDF Viewer* diferenciará al creador de la Aplicación, que adoptaría el Rol de **Presentador**, del resto de participantes del sistema, **mostrando más botones y acciones si el rol adoptado es el del Presentador**.

El motivo es el siguiente; si todos los participantes tuvieran la misma interfaz, todos los participantes podrían interactuar de la misma manera que cualquier otro participante, con la Aplicación Compartida. Sin embargo, **ésto no es deseable**.

Supongamos un caso de ejemplo: Un profesor desea impartir una clase tanto en su Aula, como a otros alumnos que se encuentran geográficamente en lugares diferentes, así que decide utilizar el sistema *Access Grid* para impartir su clase. Al impartir su clase, utilizará una presentación, así que, para que todo el mundo pueda visualizar dicha presentación, utilizará la Aplicación Compartida *Shared PDF Viewer*.

Si se deja que todos los participantes puedan interactuar de la misma manera que el profesor con la Aplicación Compartida, cualquier usuario podría, por ejemplo, cambiar de página, sin el consentimiento del profesor, o incluso cargar otro documento diferente, ya sea para realizar un acto de maldad, o simplemente por error.

Lo ideal, por tanto, sería que **el profesor sea el único que tenga control absoluto de la aplicación**, mientras que el resto, simplemente puedan visualizar el documento sin poder cambiar ni interactuar con nada de ella.

[18] De hecho, muchos visualizadores de documentos online y offline, incluso el sistema *Adobe Connect*, realizan conversiones internas a sus formatos específicos para sus visualizadores de documentos.

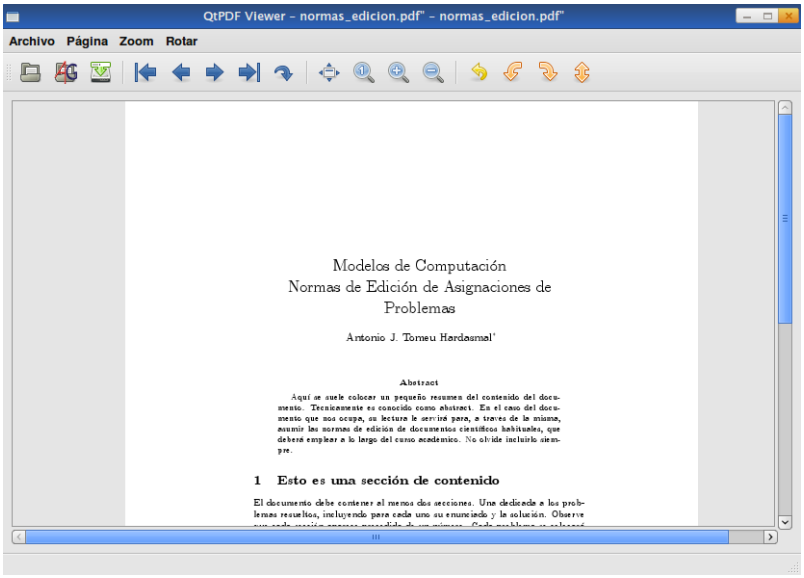


Figura 5.18: Ventana Shared PDF Viewer para el Creador

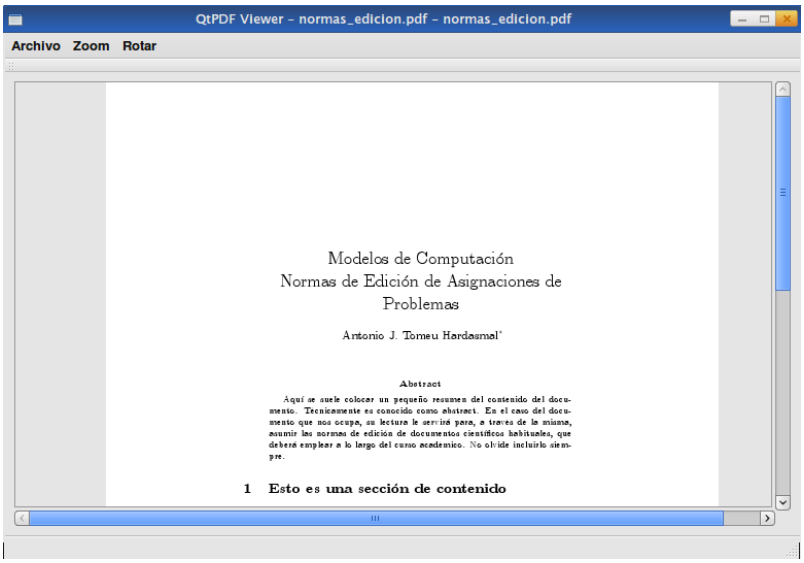


Figura 5.19: Ventana Shared PDF Viewer para el resto de participantes

Para hacer ésto, hubo que ponerse en contacto con los desarrolladores de *Access Grid*, ya que, no se encontraba ninguna función ni ninguna variable, etc, que nos devolviera quién creó una Aplicación Compartida. Efectivamente, los propios desarrolladores confirmaron que **no existe ningún elemento de la API que devuelva, de forma directa, quién creó una Aplicación Compartida**. Sin embargo, nos aportaron una solución (como siempre hacen y nosotros agradecemos su gran ayuda prestada), aunque de forma temporal, además de no ser una solución muy eficiente.

La solución temporal consiste en obtener dicha información de la propiedad `description`, al obtener el Estado de la Aplicación Compartida. Es decir:

```
1       # Obtenemos el creador de la Aplicacion Compartida
2       self._Creator = self.sharedAppClient.GetApplicationState().description
```

La variable `self._Creator`, **contendrá una cadena de texto**, del tipo ”Started by ”, más el nombre que aparece en el perfil del usuario que lo creó. Posteriormente, se acortaría dicha cadena, para quedarnos únicamente con el nombre del perfil del usuario. Luego, a la hora de generar la interfaz, se compararía esta variable con el nombre del perfil actual que se está ejecutando y, dependiendo de si coincide (en cuyo caso, sería el creador) o no, la interfaz se mostrará con más o menos acciones:

```
1       # Obtenemos el creador de la Aplicacion Compartida
2       self._Creator = self.sharedAppClient.GetApplicationState().description
3       self._Creator = self._Creator[self._Creator.find("Started by") + 11:
            self._Creator.find(" at")]
4
5       (...)
6
7       if self._Creator == self.clientProfile.name:
8           print "Es el ponente"
9           # Resto de acciones para el creador
10      else:
11          print "No es el ponente"
12          # Resto de acciones para el resto de participantes
13
14      (...)
```

Como hemos comentado, esta solución no es muy eficiente, ya que, por un lado, estamos utilizando una cadena de texto para averiguar quién es el creador, cuando lo ideal sería algún tipo de identificador (un código, etc). Por otro lado, el tener el nombre del perfil, **no siempre garantiza que sepamos quién es el creador**, puesto que puede darse la situación de que haya **dos participantes con el mismo nombre de perfil** y, por tanto, el sistema no sabría cuál de los dos es su creador. De hecho, interpretaría tal situación, como que **existen dos creadores simultáneos**.

Como estamos a la espera de una futura versión, de manera temporal, se optó por la solución aportada por el equipo de *Access Grid*.

Carga de documentos compartidos

La aplicación *Shared PDF Viewer*, cuando se selecciona el fichero a visualizar, dicho fichero **se sube al Almacén de Ficheros Compartidos del Servidor de Salas** y, automáticamente, se **envía un Evento Compartido al resto de participantes**, de forma que, cuando el resto de participantes reciban dicho Evento, se descargarán el fichero a visualizar y se mostrará en sus respectivas ventanas.

Esta acción **únicamente la podrán realizar el creador de la Aplicación Compartida**, es decir, el resto de participantes no podrán cargar ningún documento.

Cambio de página

Otro Evento Compartido creado **se produce al cambiar de página**, ya que, cuando el usuario cambie de página, automáticamente, la Aplicación Compartida, notifica al sistema *Access Grid* que se ha producido dicho evento y se lo transmiten al resto de participantes, **actualizando a la página cambiada**.

Este evento es muy útil, ya que es quien permite mantener el flujo de la presentación y que el resto de participantes puedan visualizarla y seguirla.

Esta acción **únicamente la podrán realizar el creador de la Aplicación Compartida**, es decir, el resto de participantes no podrán cambiar de página en ningún momento.

¿Qué pueden hacer el resto de participantes?

Aunque las acciones principales de un visor de documentos PDF están limitadas únicamente al creador de la Aplicación Compartida, no significa que el resto de participantes no puedan realizar ningún tipo de acción. A continuación se enumeran las acciones disponibles para ellos (y, por tanto, para el propio creador también):

- Posibilidad de almacenar una copia local del documento cargado en su disco duro.
- Posibilidad de cambiar el tamaño de visualización del documento, haciendo zoom, bien para ampliar, bien para disminuir, **incluso puede visualizar el documento a Pantalla Completa**, y el resto de usuarios visualizarlos de formas diferentes.
- Posibilidad de rotar el documento a cualquier dirección deseada.

5.7.2. Versión Final utilizando *Qt4*

Inicialmente, ya que todas las aplicaciones, tanto internas, como compartidas, del sistema *Access Grid*, sus interfaces están desarrolladas bajo *wxPython*, definitivamente, por motivos de **no poder ofrecer, bajo dicha plataforma, esta Aplicación Compartida en sistemas Windows**, tuvo que rehacerse parte de la aplicación, para desarrollarla bajo *Qt4*.

El motivo es muy simple, la renderización de documentos PDF la realiza una librería llamada **`Poppler`**. Esta librería es muy conocida ya que se utiliza para la gran mayoría de visores PDF libres existentes [19], [20].

Para *Python* se han desarrollado unos *bindings* permitiendo, por tanto, poder renderizar documentos PDF en aplicaciones escritas en *Python*. Dicha librería se llama **`Python Poppler (o bien PyPoppler)`** [21].

El problema radica en que dicha librería, a fecha de hoy, **únicamente está disponible solamente para sistemas Linux**, por lo que, impedía que la aplicación *Shared PDF Viewer* pudiese ser multiplataforma. Un grupo de usuarios consiguieron compilar el código fuente de estos bindings y, por tanto, **generar las librerías para Windows**, sin embargo, tenía la limitación de que era necesario utilizar *Python 2.6*, cuando, hay que recordar, que *Access Grid* está desarrollado bajo *Python 2.4*.

Después de muchas búsquedas de información, se descubre que **otro equipo de desarrollo han realizado otros bindings para *Python*** de la librería *Poppler*, pero adaptadas, en exclusiva, al entorno *Qt*. Esta librería se llama ***PyPoppler-Qt4*** [22].

Así pues, gracias a esta librería, fue posible hacer la aplicación *Shared PDF Viewer* multiplataforma.

[19] `http://poppler.freedesktop.org/`
[20] `http://es.wikipedia.org/wiki/Poppler_(biblioteca)`
[21] `https://launchpad.net/poppler-python`
[22] `https://code.launchpad.net/~spv-dev/slidepresenterview/pypoppler-qt4`

Inconvenientes

Aunque gracias a esta librería, fue posible hacer la aplicación multiplataforma, plantea una serie de inconvenientes:

- La API de las *Aplicaciones Compartidas en Access Grid* está adaptada para wxPython, por lo que hay que, primero crear una aplicación tipo wxPython e, internamente, crear la aplicación bajo Qt4. Ésto plantea un problema de optimización, ya que no se están utilizando los recursos necesarios.
- Realizar una aplicación en Qt4 implica **distribuir las librerías de Qt4 junto a la aplicación**. Esto hace que la aplicación ocupe más espacio físico. El sistema *Access Grid* ya incorpora las librerías de wxPython, por lo que es una tarea que el desarrollador de Aplicaciones Compartidas se ahorraría a la hora de distribuir.
- La calidad del renderizado no es tan buena como la librería *PyPoppler*, ya que, una vez que se hace zoom, **pixela un poco la visualización**, dificultando un poco la visualización de textos con letra pequeña. Sin embargo, hay que aclarar que, dicha pixelación se mantiene al hacer sucesivos zooms.

Aún existiendo estos inconvenientes, se **prefirió disponer de una Aplicación Compartida multiplataforma**, debido a la diversidad de sistemas que los participantes utilizan.

5.7.3. Análisis y Diseño de la aplicación Shared PDF Viewer

A continuación se explicará, brevemente, el análisis y diseño que se realizó antes de desarrollar la aplicación *Shared PDF Viewer*.

Se ha procurado que las Aplicaciones Compartidas que se han desarrollado para *Access Grid* tengan un diseño similar.

Primeramente, se desarrolla la aplicación de manera ”independiente”, es decir, como si no fuera a formar parte del sistema *Access Grid*. Se ha intentado realizar el patrón **Modelo Vista Controlador**, así pues, la interfaz está implementada en un módulo independiente. Luego se ha creado un **Controlador** que contiene todas las funciones principales sobre manejos de documentos PDF. En cuanto al modelo, realmente, no existe un modelo de datos, puesto que no hay ningún dato que tratar, únicamente abre documentos PDF desde el equipo y lo manipula.

Una vez desarrollada la aplicación, ésta **es adaptada para integrarla al sistema *Access Grid***. Para ello, se crea una **clase que herede la interfaz** y se sobrecargan los eventos, creando, por tanto, los Eventos Compartidos, realizando, prácticamente, las mismas acciones, salvo que éstas se comunican al resto de participantes.

Diagrama de Clases

Basándonos en lo explicado anteriormente, el diagrama de clases realizado de la aplicación *Shared PDF Viewer* es el siguiente:

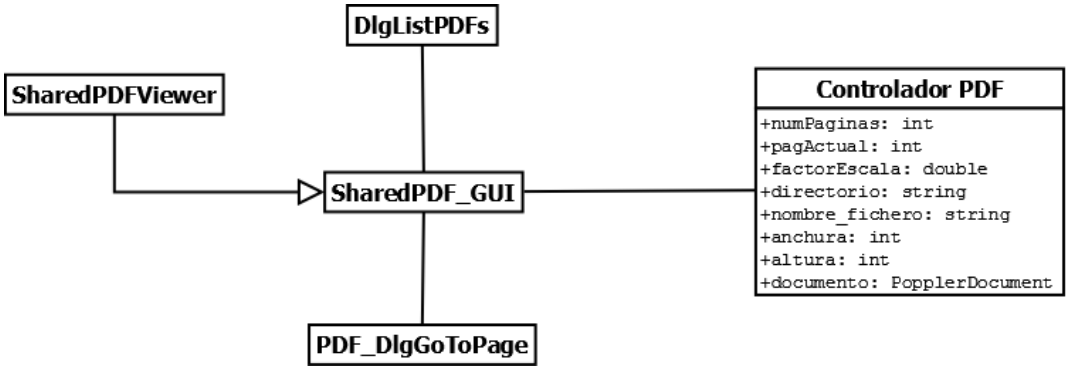


Figura 5.20: Diagrama de Clases de Shared PDF Viewer

Diagramas de Secuencia

A continuación, se detallan los diferentes Diagramas de Secuencia, mostrando las acciones de cada una de las acciones de la aplicación:

- **Abrir un Documento:**

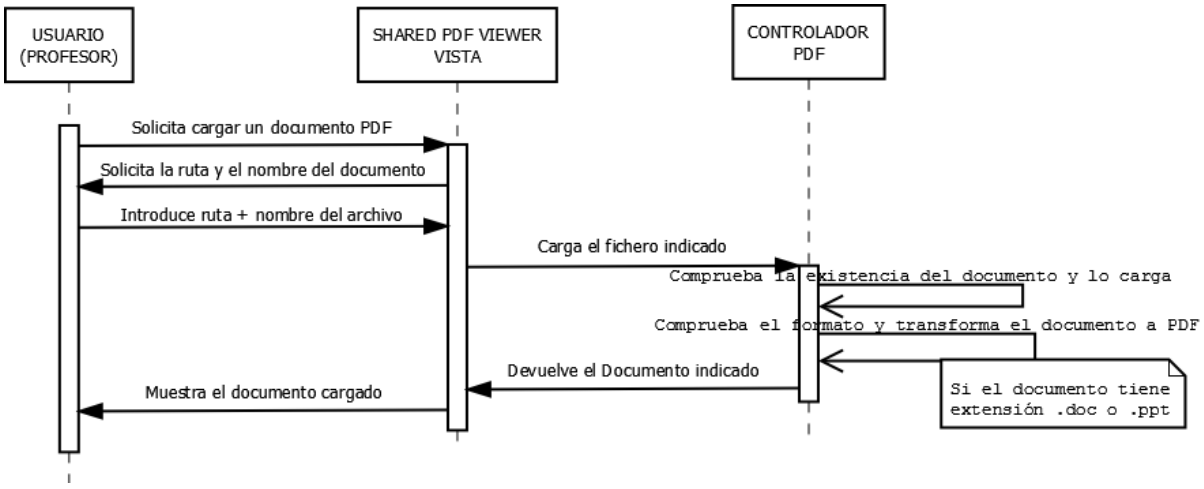


Figura 5.21: Diagrama de Secuencia - Abrir Documento

- **Guardar un Documento:**

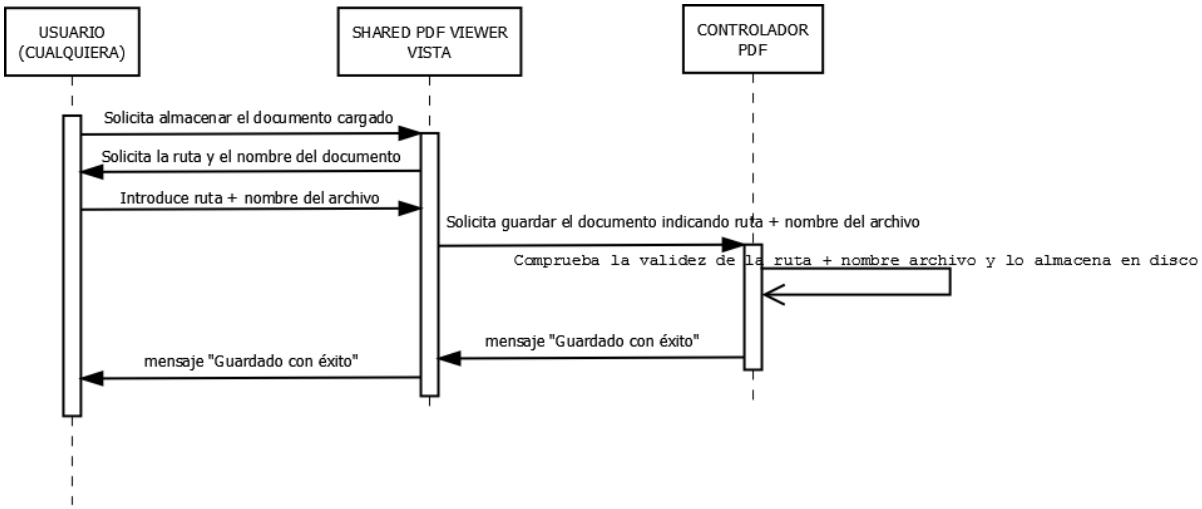


Figura 5.22: Diagrama de Secuencia - Abrir Documento

- **Ir a la Primera Página:**

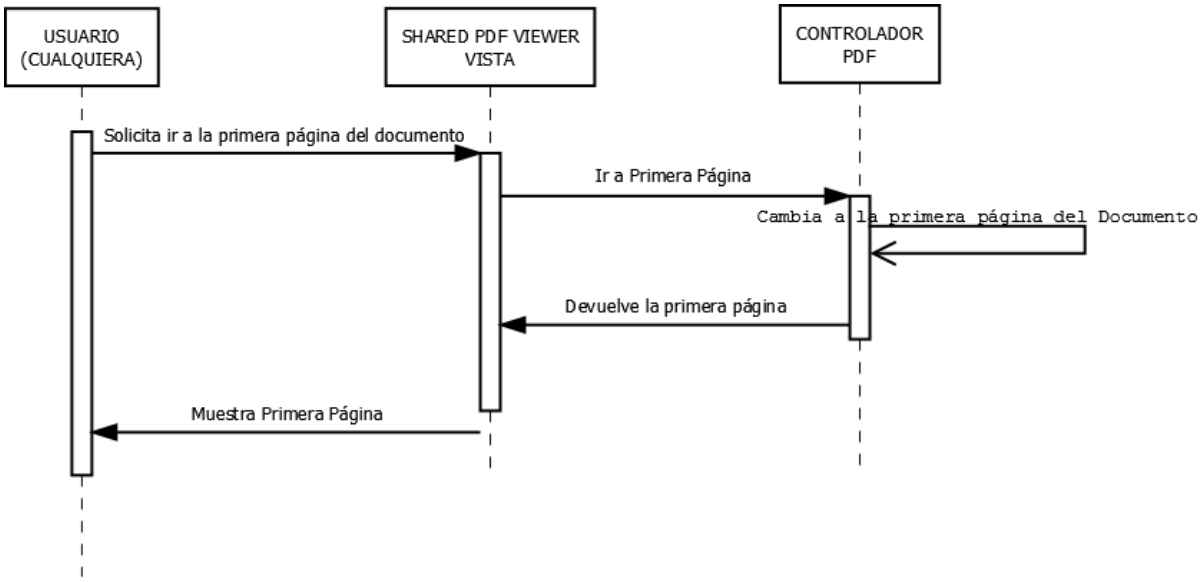


Figura 5.23: Diagrama de Secuencia - Abrir Documento

- **Ir a la Última Página:**

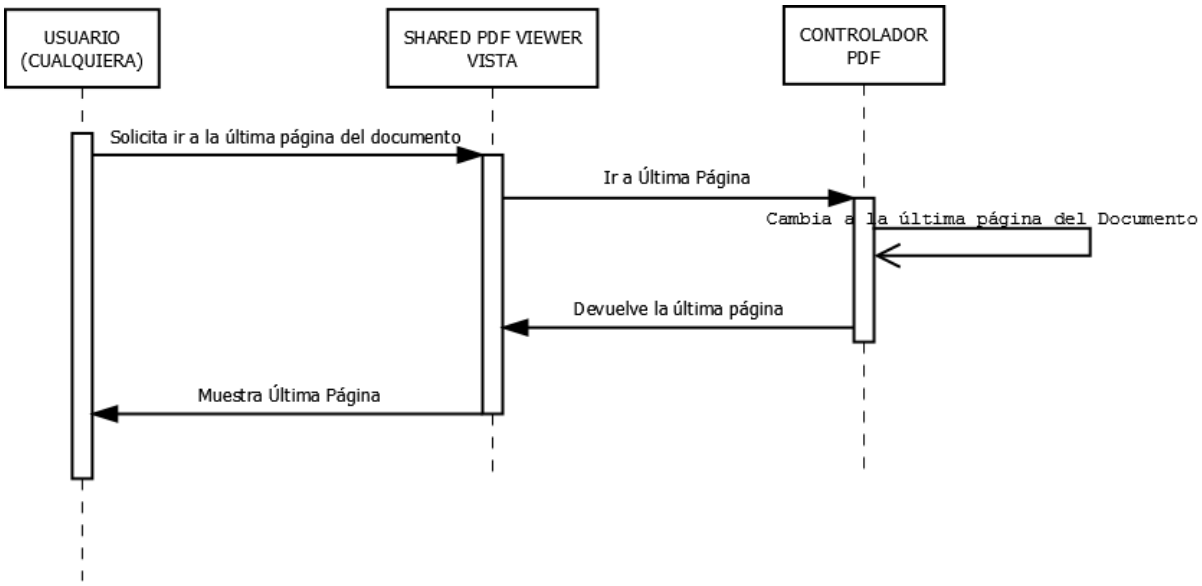


Figura 5.24: Diagrama de Secuencia - Abrir Documento

- **Ir a la Página Siguiente:**

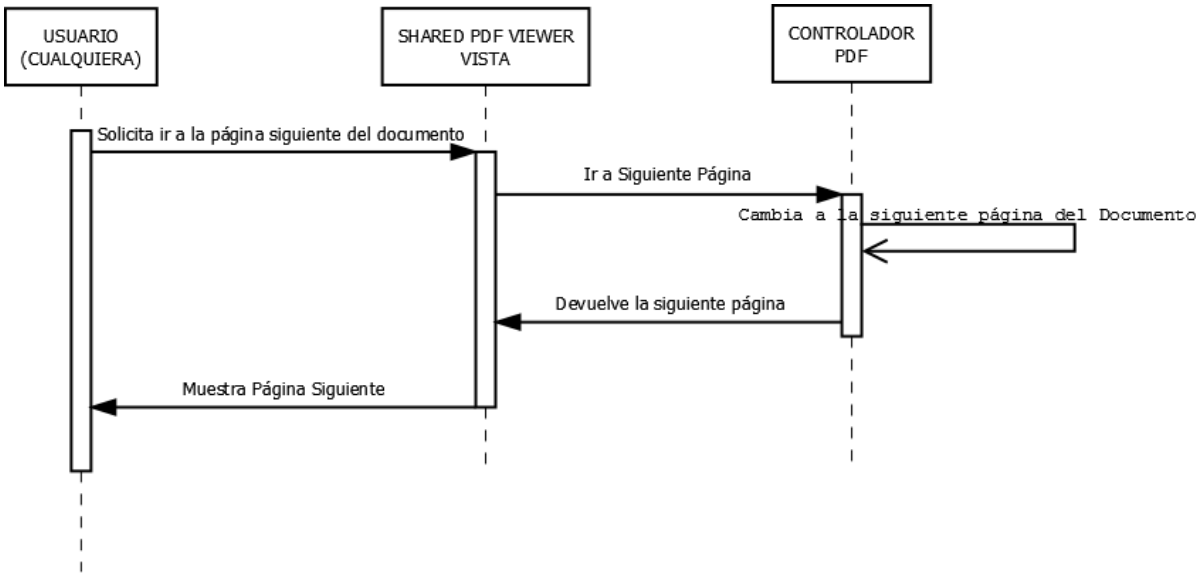


Figura 5.25: Diagrama de Secuencia - Abrir Documento

- **Ir a la Página Anterior:**

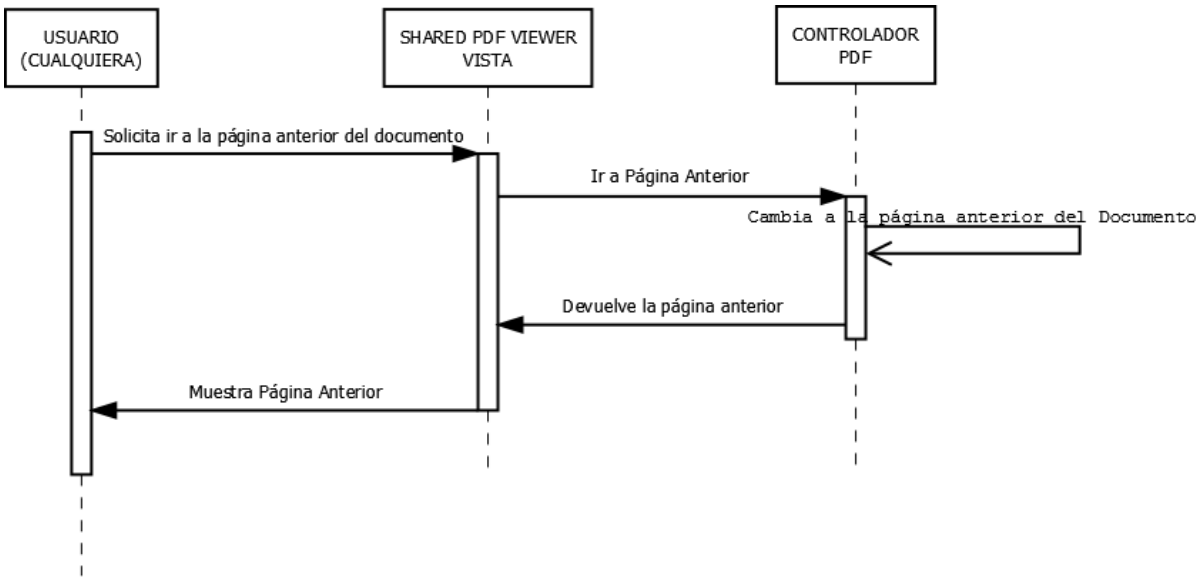


Figura 5.26: Diagrama de Secuencia - Abrir Documento

5.7.4. Código fuente de la aplicación Shared PDF Viewer

El resultado final de la aplicación ***Shared PDF Viewer*** se encuentra disponible en la siguiente dirección web:
`https://forja.rediris.es/plugins/scmsvn/viewcvs.php/SharedApplications/SharedPDFViewer-Final/?root=cusl4-aguca`.

5.8. Diseño y realización de cuestionarios

Siguiendo con el desarrollo de Aplicaciones Compartidas, se pensó en realizar una aplicación para diseñar cuestionarios y se pudieran resolver, procurando que el procedimiento sea lo más rápido y cómodo posible tanto para el profesor como el alumno. Esta aplicación se llama ***Shared Evaluation***.

La idea de desarrollar esta aplicación surge al utilizar el sistema *Adobe Connect*, ya que, entre las aplicaciones integradas que dispone, está un diseñador de cuestionarios.

La idea es muy simple. Existen dos aplicaciones, una para el profesor, y otra para los alumnos, por tanto, una vez más, dependiendo de quién sea el creador, se le ejecutará una aplicación u otra. Si es el profesor, se le mostrará un listado de tests ya creados, junto a un conjunto de botones para poder crear un nuevo test, editar uno existente o borrarlo. Además, permite **publicar un test**, entendiéndose ésto como que dicho test ya está disponible para que los alumnos lo resuelvan.

Tanto si elige crear un nuevo test, como si elige editar uno ya existente, se le cargará un formulario tal como éste:

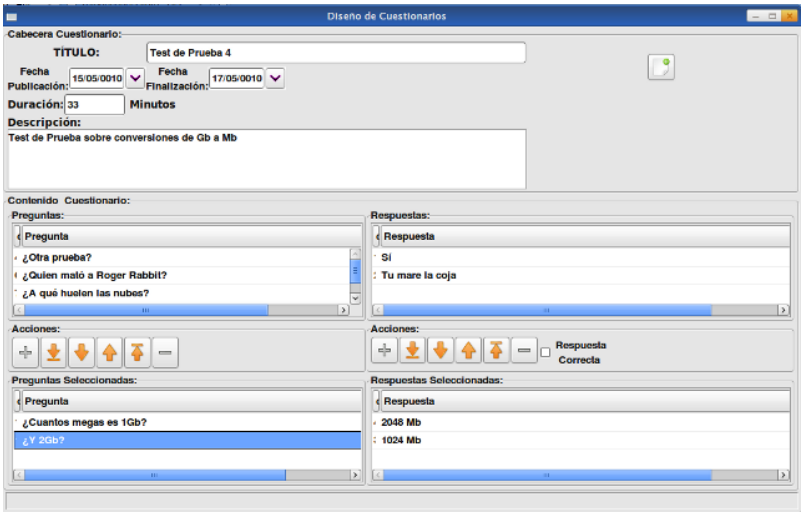


Figura 5.27: Aplicación Compartida - Shared Evaluation

Se puede observar que la interfaz está dividida en tres grandes bloques. El primero de ellos, contiene el conjunto de datos referente al propio test, tales como un *título*, la *fecha de publicación*, o una *descripción*.

El segundo bloque hace referencia al conjunto de preguntas del test. Este bloque, a su vez, está dividido en dos partes, la superior, contiene un conjunto de preguntas ya creadas. El profesor tiene a su disposición un **catálogo de preguntas disponibles** y, en caso de que no encuentre la pregunta deseada, puede crearse una propia y se añadirán al catálogo de preguntas disponibles. El profesor, selecciona una pregunta y lo añade a su test, indicando que dicha pregunta formará parte del test. Así, el profesor iría creando su test, indicando qué preguntas lo formarán.

El tercer, y último bloque, hace referencia al conjunto de respuestas de cada Pregunta del test. Este bloque, una vez más, está dividido en dos partes, la superior, contiene un conjunto de respuestas ya creadas. El profesor tiene a su disposición un **catálogo de respuestas disponibles** y, en caso de que no encuentre la respuesta deseada, puede crearse una propia y añadirla al catálogo de respuestas disponibles. El profesor selecciona una de las preguntas que forman parte del test, a continuación, seleccionará cada respuesta que quiere que forme parte de dicha pregunta, indicando, además, si dicha respuesta es correcta o no en ese test. Una vez haya indicado todas las preguntas y sus respuestas, indicando cuáles son las correctas, el profesor finalizará de crear el test.

Tanto los tests, como el conjunto de Preguntas y Respuestas, están almacenadas en una **base de datos**, en nuestro caso, elegimos **MySQL**.

Si el usuario es un alumno, como se ha comentado, se ejecutará otra Aplicación Compartida. En este caso, se le aparecerá un listado de **tests disponibles**, es decir, aquellos tests que el alumno puede realizar en ese momento. El alumno elegirá un test y pulsará un botón para comenzar a realizarlo, mostrándole una nueva ventana, con el test diseñado. Se le mostrará un listado de Preguntas y las Respuestas que las componen. Además, tendrá una cuenta atrás (según la duración indicada a la hora de crear el test). El alumno indicará la o las respuestas que considere que son correctas y enviará el test resuelto. Por último, el sistema *Access Grid*, evaluaría los tests recibidos, notificando la nota al alumno.

Realmente, como puede observar, **no existen Eventos Compartidos** en esta Aplicación Compartida,

únicamente cuando se diferencia quién es el profesor y quiénes los alumnos. Sin embargo, es una muestra de que **se pueden realizar Aplicaciones Compartidas, utilizando una Base de Datos e integrarla en *Access Grid***.

5.8.1. Análisis y Diseño de la aplicación Shared Evaluation

Diagrama de Clases

El diagrama de clases realizado de la aplicación *Shared Evaluation* es el siguiente:

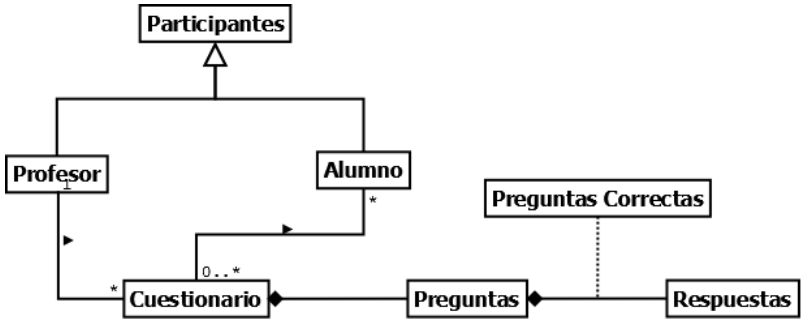


Figura 5.28: Diagrama de Clases de Shared Evaluation

Diagrama Entidad/Relación

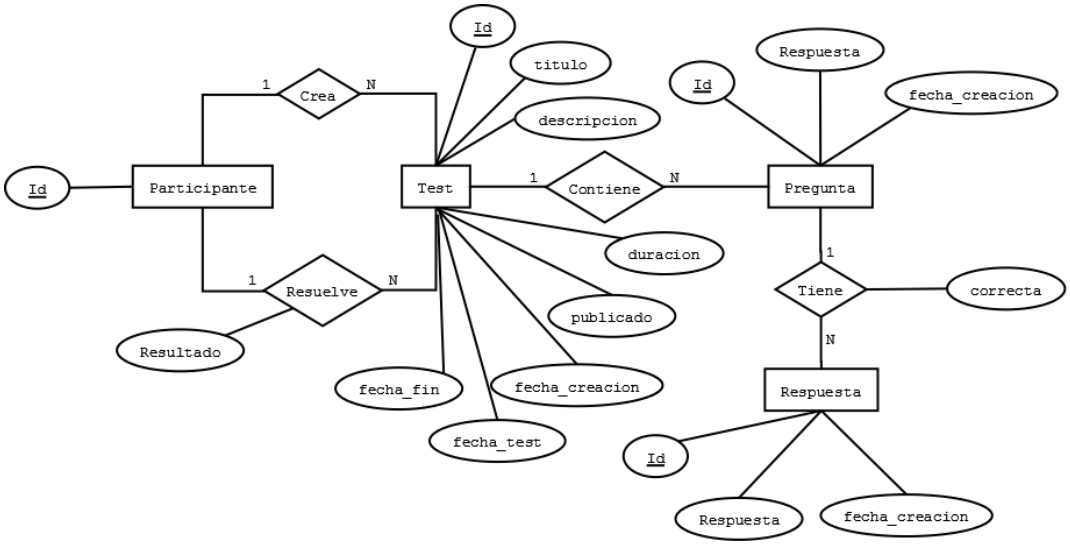


Figura 5.29: Diagrama de E-R de Shared Evaluation

Diagramas de Secuencia

A continuación, se detallan los diferentes Diagramas de Secuencia, mostrando las acciones de cada una de las acciones de la aplicación:

- **Crear un test:**

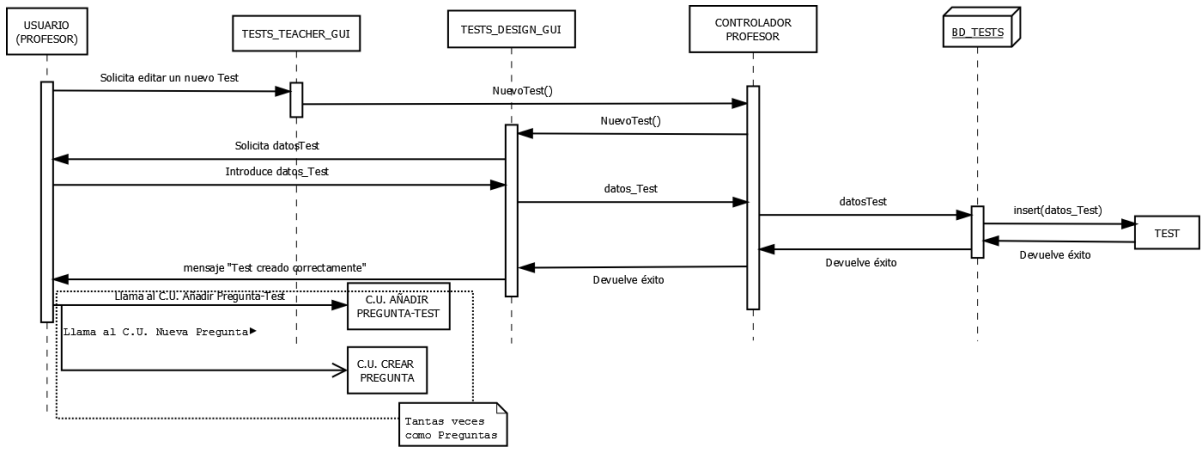


Figura 5.30: Diagrama de Secuencia - Nuevo test

- **Añadir Pregunta a un test:**

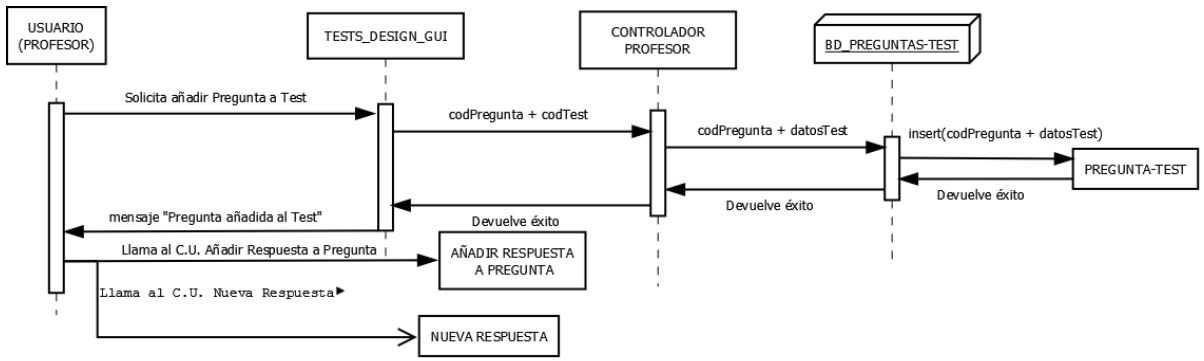


Figura 5.31: Diagrama de Secuencia - Añadir Pregunta a un test

- **Añadir Respuesta a una Pregunta del test:**

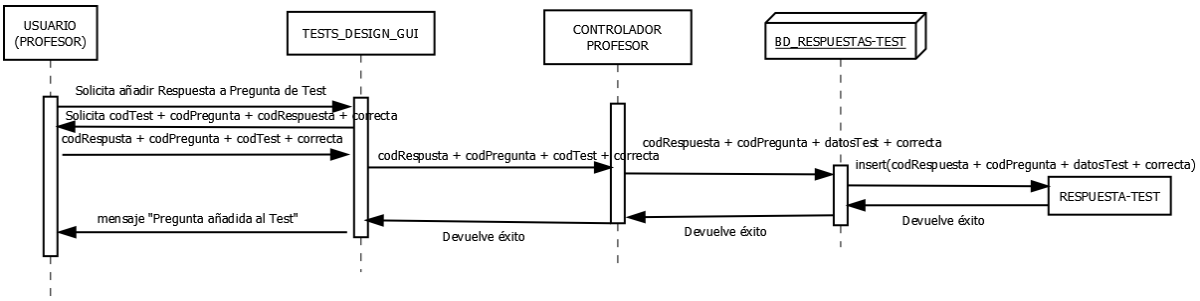


Figura 5.32: Diagrama de Secuencia - Añadir Respuesta a una Pregunta del test

- **Crear Pregunta:**

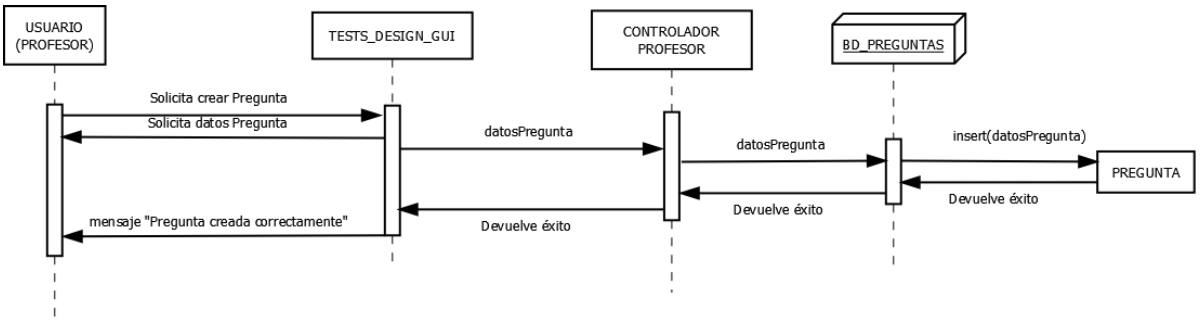


Figura 5.33: Diagrama de Secuencia - Crear Pregunta

- **Crear Respuesta:**

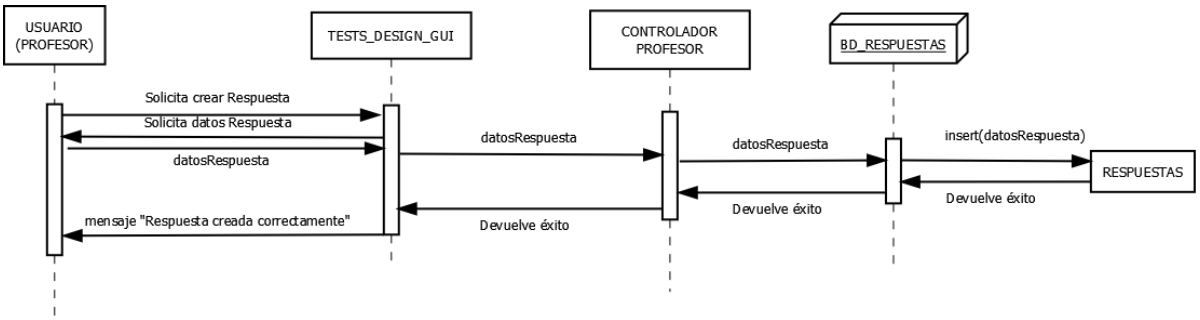


Figura 5.34: Diagrama de Secuencia - Crear Respuesta

- **Editar test:**

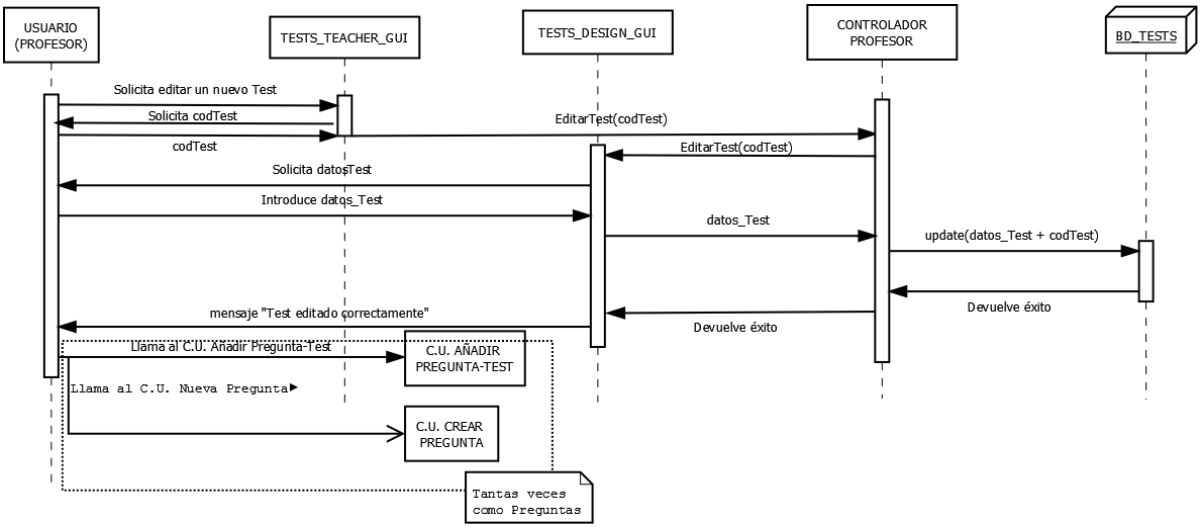


Figura 5.35: Diagrama de Secuencia - Editar test

- **Borrar test:**

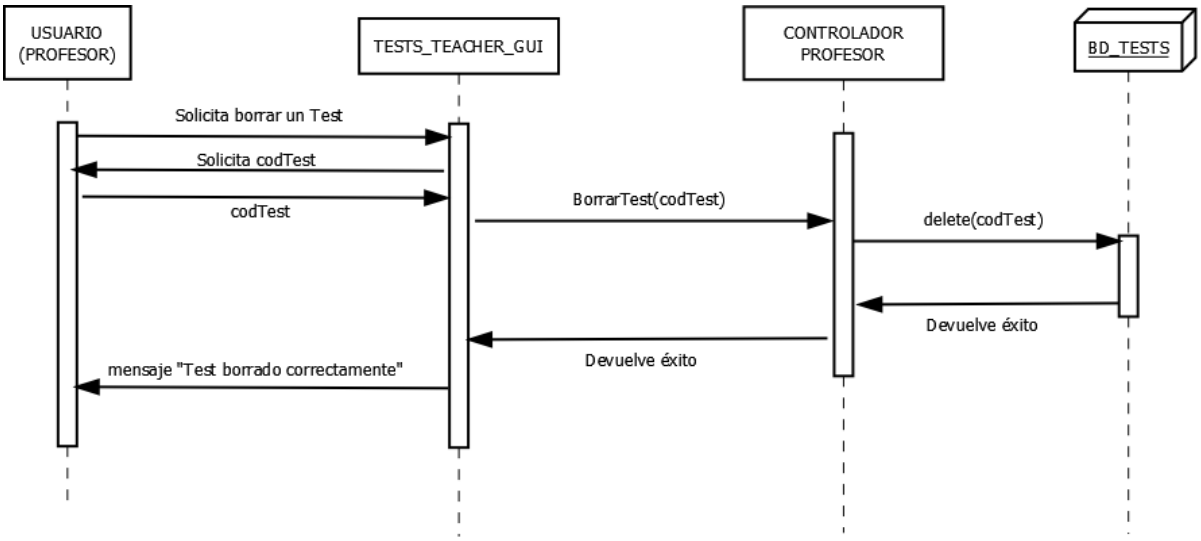


Figura 5.36: Diagrama de Secuencia - Borrar test

- **Borrar Pregunta:**

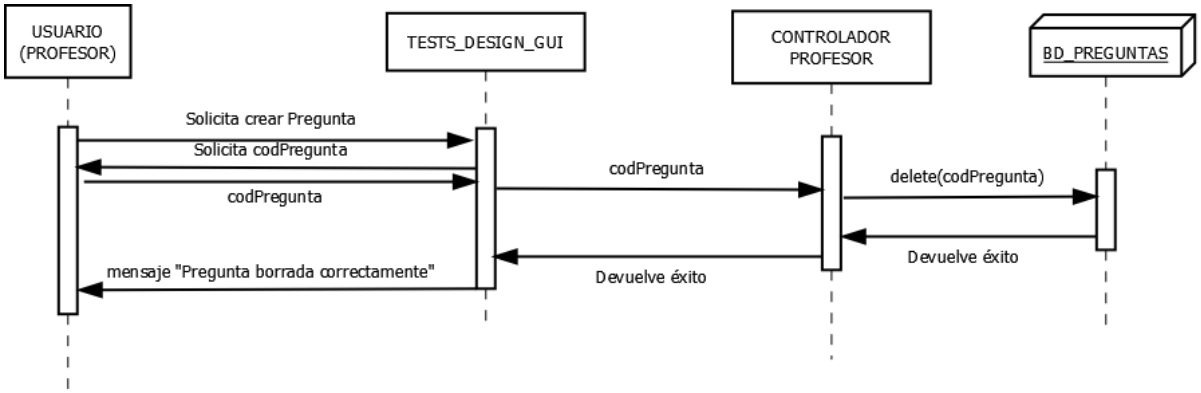


Figura 5.37: Diagrama de Secuencia - Borrar Pregunta

- **Borrar Respuesta:**

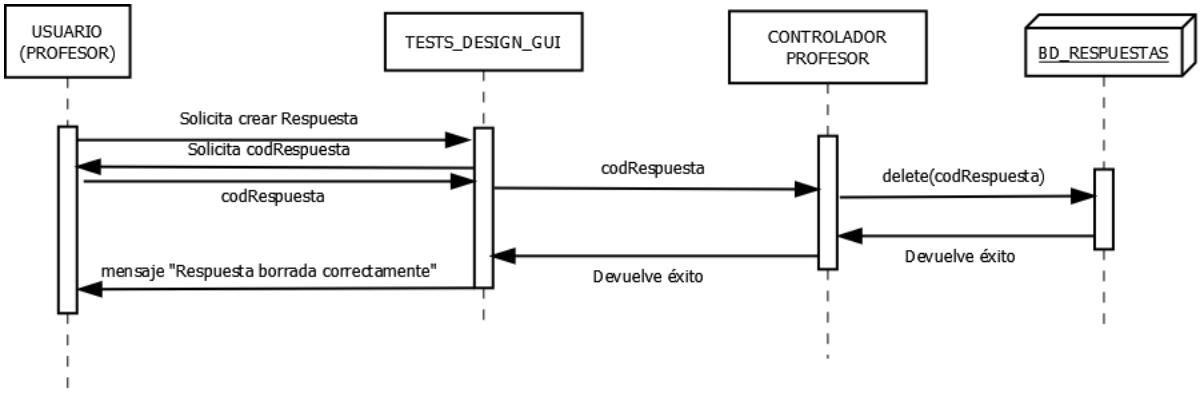


Figura 5.38: Diagrama de Secuencia - Borrar Respuesta

5.8.2. Código fuente de la aplicación Shared PDF Viewer

El resultado final de la aplicación ***Shared Evaluation*** se encuentra disponible en la siguiente dirección web: `https://forja.rediris.es/plugins/scmsvn/viewcvs.php/SharedApplications/SharedEvaluation/?root=cusl4-aguca`.

5.8.3. Aplicación Sin Terminar

Como se puede comprobar, el Código fuente mostrado está incompleto, ya que falta por desarrollar algunas cosas. Debido a los problemas surgidos a lo largo del desarrollo de esta beca, junto a que el personal de la Universidad de Cádiz **determinaron otros objetivos prioritarios**, el desarrollo de esta Aplicación Compartida **se detuvo**.

El personal de la Universidad de Cádiz conocen el desarrollo de esta aplicación y, aunque les parecen interesante, consideraban oportuno realizar antes otros objetivos pendientes, y establecer éste como **futuro objetivo**.

Desgraciadamente no se pudo finalizar el desarrollo, aunque **gran parte está desarrollada**, quedando pendiente de finalizar la parte correspondiente al alumno, que, únicamente, se consiguió generar un test a partir de la Base de Datos:

Figura 5.39: Shared Evaluation - Test Generado

Sin embargo, aún no es posible guardar las respuestas marcadas y, por tanto, no se pueden enviar para su evaluación.

Se espera que, pronto, se retome el desarrollo de esta aplicación, para finalizarla completamente.

5.9. Otros Desarrollos

Además de corregir errores del sistema *Access Grid*, desarrollar nueva Aplicaciones Compartidas y añadir nuevas funcionalidades, se han hecho otros desarrollos en relación a *Access Grid*.

5.9.1. Mejora del Código de Access Grid

Como último campo de desarrollo a ”visitar”, se investigó sobre cómo mejorar el sistema *Access Grid*. Al igual que la Aplicación Compartida anterior, debido a la falta de tiempo no se ha podido mejorar muchos aspectos del sistema, **aunque se han establecido, como futuros objetivos, añadir nuevas funcionalidades al sistema *Access Grid***.

Sin embargo, sí dio tiempo a investigar un poco cómo añadir una nueva funcionalidad y se hizo un pequeño ejemplo. Concretamente, entre los futuros objetivos, estaba el **añadir un nuevo botón para guardar el historial del chat**.

Este botón era ideal que se encontrase en la interfaz principal del sistema *Access Grid*, junto al propio chat de la interfaz. Así pues, para añadir esta nueva funcionalidad, era necesario corregir el código fuente de dicha interfaz, éste es, el fichero *VenueClientUI.py*.

Guardar el chat en un fichero de texto fue sumamente fácil, ya que, el componente que utilizaron para mostrar el chat era un componente de tipo **Memo**. En cualquier entorno de desarrollo de interfaces

de usuario, los componentes Memo, suelen incluir un **método que permite almacenar su contenido en un fichero**. Así pues, concretamente, bajo wxPython, el método es `SaveFile`.

Así pues, lo único que hizo falta fue, por un lado, crear un nuevo botón, cuyo texto es "Guardar", y, ademas, su correspondiente evento al hacer click, que preguntará en qué fichero desea almacenar el historial del chat y lo almacenará.

Para ello se hizo lo siguiente en el fichero *VenueClientUI.py*:

- En la clase `class`, dentro de la función de iniciación, `__init__`, se añadieron las siguientes líneas, justo después de declarar el componente cuadro de texto, que es donde el usuario escribe en el chat. Con esto, creamos el nuevo botón, además de declarar su correspondiente método, al hacer click en él:

```
class JabberClientPanel(wx.Panel):

    ID_BUTTON = wx.NewId()
    ID_WINDOW_TOP = wx.NewId()
    client =''

    def __init__(self, parent, id):
(...)
################################################
### MEJORA DE ACCESS GRID - GUARDAR HISTORIAL ###
################################################
        # Boton para guardar el historial del chat en un fichero
        self.ID_SAVECHAT = wx.NewId()
        self.cmdSaveChat = wx.Button(self.panel, self.ID_SAVECHAT,
            label='Guardar',
            pos=(8, 8), size=(80, 28))
        wx.EVT_BUTTON(self, self.ID_SAVECHAT, self.SaveChat)
################################################
###                                          ###
################################################
```

- Justo después de finalizar la función de inicialización `__init__`, añadimos la siguiente función, que corresponde al método asociado al hacer click en el botón creado:

```
################################################
### MEJORA DE ACCESS GRID - GUARDAR HISTORIAL ###
################################################
    def SaveChat(self, event):
        """ Guarda, en un fichero, el historial del chat """
        log.debug("Guardando historial en fichero")

        dlgSave = wx.FileDialog(self, "Guardar chat de Access Grid", "
            ", "", "*.txt", wx.SAVE)
        if dlgSave.ShowModal() == wx.ID_OK:
            path = dlgSave.GetPath()
            self.textOutput.SaveFile(path)
            log.debug("Fichero guardado como: " + path)

        dlgSave.Destroy()
```

```
15  ################################################
16  ###                                          ###
17  ################################################
```

- Por último, guardamos el fichero, y ejecutamos el Cliente de *Access Grid*, para notar los resultados. Al conectarnos a una Sala, debería de poderse visualizar el botón, y comprobar que, efectivamente, funciona:

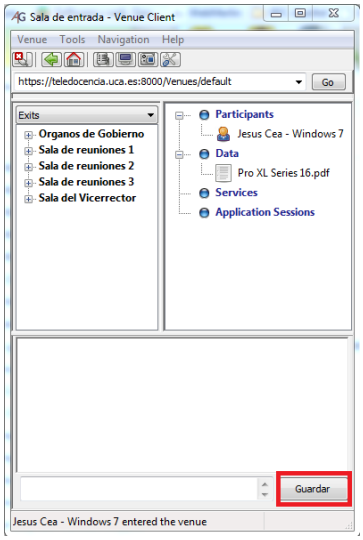


Figura 5.40: Cliente Access Grid con el nuevo botón

El fichero fuente al completo se encuentra disponible en la siguiente dirección web: `https://forja.rediris.es/plugins/scmsvn/viewcvs.php/MejorasCodigo/BotonGuardarChat/?root=cusl4-aguca`.

5.9.2. Plugin AgridLauncher

Como solicitud por parte del personal de la Universidad de Cádiz, se ha desarrollado un **Plugin para Access Grid** cuya finalidad es **entrar en una Sala desde un enlace web**.

Es decir, una vez instalado el Plugin, cuando se visite una página web con su navegador preferido, si dicha página web contiene enlaces de Salas de *Access Grid*, al hacer click sobre ellas, automáticamente se ejecutará el Cliente de *Access Grid* y entrará automáticamente en la Sala cliqueada.

Para desarrollar este Plugin, se basó, principalmente, en aquellos programas que se ejecutan con ciertos enlaces web, tales como, por ejemplo, los **enlaces ed2k**, que, al hacer click en un enlace web cuyo protocolo sea `ed2k://`, automáticamente se ejecuta la aplicación *eMule* y se añade como descarga el fichero que corresponde con el enlace seleccionado. Existen muchas aplicaciones con sus protocolos personalizados tales como *rFactor*, *Skype*, *Gaim*, etc.

Así que lo que se hizo fue **crearnos nuestro propio manejador de protocolo**, llamado *agrid*. De forma que, cuando una página web contiene enlaces que comienzan por el protocolo *agrid://*, automáticamente se ejecutará el Cliente de *Access Grid*, entrando en la Sala indicada por dicho enlace.

Crear un manejador de protocolo es sencillo, tanto en Windows, como en Linux. Simplemente se crea una nueva entrada en el registro.

Windows

La estructura del registro de Windows debería ser como la siguiente:

```
HKEY_CLASSES_ROOT
    alert
      (Default) = "URL:Agrid Protocol"
      URL Protocol = ""
      shell
      open
         command
               (Default) = C:\Python24\pythonw.exe
                "C:\Archivos de Programa\AGTk-3\bin\agridLauncher.pyw" %1
```

Donde `agridLauncher.pyw` es un script desarrollado para que *Access Grid* soporte el protocolo *agrid*, y "%1" es un parámetro, que en nuestro caso sería la dirección de una Sala.

Linux

Para hacer lo mismo en sistemas Linux, escribimos los siguientes comandos en una Terminal:

```
gconftool-2 --set --type=string /desktop/gnome/url-handlers/agrid/command
        '/usr/bin/agLauncher.sh %s'
gconftool-2 --set --type=boolean /desktop/gnome/url-handlers/agrid/enabled true
gconftool-2 --set --type=boolean /desktop/gnome/url-handlers/agrid/needs_terminal false
```

Por último, el script al que hace referencia, únicamente quita del enlace el protocolo *agrid://* y llama a la aplicación *VenueClient* pasándole, como argumento (con la opción −−url), la dirección obtenida del enlace.

5.9.3. Gestor de contenidos

Otro desarrollo muy importante lo realizó mi compañero de beca Miguel Ángel Pérez Prado que, además, es una **parte muy importante de su Proyecto de Final de Carrera**. Debido a ésto, no se entrará en detalle sobre este desarrollo, pero se comentará con el fin de que se pueda observar el gran desarrollo y el gran soporte que se le está ofreciendo tanto al sistema *Access Grid*, como a las Salas de Teledocencia en sí.

Uno de los objetivos primarios establecidos en la primera reunión era el **poder grabar las sesiones** en vídeo digital. Es decir, todo acto que se celebrase en una Sala de Teledocencia, debe de haber la posibilidad de poder grabarlo en un fichero y, además, poder distribuirlo de forma rápida.

Miguel ha estado trabajando duro en este aspecto y ha conseguido grabar las sesiones. Para distribuir los vídeos, ha desarrollado una aplicación web, donde, además de contener un foro para preguntar dudas, aportar ideas, opiniones, etc; una galería de imágenes y una zona de descargas, donde se encuentran las aplicaciones desarrolladas; se encuentra una **galería de vídeo**, donde contiene todos los vídeos que se vayan grabando conforme se vayan desarrollando sesiones de *Access Grid*.

Miguel detallará todo este desarrollo con mucho más detalle al presentar su Proyecto Final de Carrera.

Capítulo 6

Conclusiones

6.1. Conclusiones del Proyecto

Desarrollar este Proyecto de Final de Carrera ha sido **toda una experiencia muy positiva** por muchos motivos. Aunque es cierto que ha habido momentos de desesperación muy frustrantes, sólo representa una minoría con respecto al lado positivo de esta experiencia vivida.

Se han aprendido muchas cosas nuevas, desde programar con un lenguaje de programación nunca utilizado, como es *Python*, pasando por utilizar bibliotecas como *Qt4* o *wxWidget*, conocer nuevos sistemas como ha sido el propio *Access Grid* y *Adobe Connect*, y terminando por el aprendizaje de utilizar el equipamiento audiovisual con la última tecnología.

La implantación del sistema *Access Grid* en las Salas de Teledocencia ha sido todo un éxito, quedando muy satisfechos tanto el personal de la Universidad de Cádiz, como el personal de la empresa Auditel. Es cierto que hubo muchas complicaciones, principalmente, por la falta de conocimiento tanto del sistema *Access Grid*, como del equipamiento existente en las Salas. También se dedicó bastante tiempo en intentar conseguir la mejor calidad de vídeo, trabajando constantemente con el códec *MPEG-4* e investigando por qué se pixelaban las fuentes de vídeo. Sin embargo, mirando el lado positivo de estos inconvenientes, es que, al intentar conocer cada una de sus opciones y funciones, en busca de la solución al problema de pixelación, se consiguió una mejor formación en el sistema *Access Grid*.

El desarrollo para *Access Grid* ha sido también toda una experiencia. Además de aprender el lenguaje *Python*, hubo que documentarse mucho para poder comprender lo mejor posible la API proporcionada por el equipo de desarrollo de *Access Grid*. Entre corregir errores de su código fuente, mejorar el sistema *Access Grid*, añadiendo nuevas funciones y desarrollar Aplicaciones Compartidas para él, ésta última sería la preferida por el autor de este Proyecto Final de Carrera. Los resultados que se obtienen son muy satisfactorios y, el potencial que ofrece la API, hace al desarrollador pensar constantemente en nuevas ideas para dessarrollar.

Pero además de la satisfacción de desarrollar una Aplicación Compartida, también está la satisfacción del software desarrollado. Por ejemplo, nunca antes se había trabajado con librerías para renderizar documentos en PDF y conseguir desarrollar un visor de documentos en PDF y ver cómo funcionaba fue una experiencia bastante gratificante.

Por otro lado, hay que destacar la complejidad que ha habido tanto para corregir errores para *Access Grid*, como para añadir nuevas funcionalidades. Los motivos son varios; el sistema es muy complejo y contiene una gran cantidad de ficheros, por lo que, saber a priori, por dónde puede venir el fallo no es tarea nada fácil, pero creo, que el principal motivo es **la falta de experiencia**. Es la primera vez que se

coge un sistema hecho por otras personas, ver su código fuente y corregir su código fuente para solucionar problemas, además de añadir nuevas funciones. Por lo que no se está habituado a esta forma de trabajar y, por eso, aumentó la complejidad del proyecto.

Sin embargo, lo que más se podría destacar como resultado satisfactorio, es el **esfuerzo personal aplicado** para que, partiendo de un desconocimiento casi absoluto tanto del sistema *Access Grid*, como del equipamiento completo de las Salas, se haya conseguido una experiencia que, a día de hoy, se demuestra bastante soltura. Además, si ocurriera algún error inesperado, somos capaces de identificar el motivo del error y solventarlo lo antes posible. Todos los Campus de la Universidad de Cádiz tienen instalado un cliente de *VNC*, de forma que, si necesitan algún tipo de ayuda o si tienen algún problema y, a través de nuestras indicaciones no consiguen solucionarlo, tomamos remotamente el control de sus equipos para que nosotros intentemos solucionarlos.

Es una clara puesta en práctica de lo que un Ingeniero debería ser capaz de hacer: **enfrentarse a un nuevo problema y saber solucionarlo**.

Por último, destacar, como muestra de la satisfacción del personal tanto de la Universidad de Cádiz, como de Auditel, de los resultados conseguidos, existen varios objetivos futuros que se desean desarrollar para reforzar más aún el apoyo a las Salas de Teledocencia, implicando, por tanto, la posibilidad de una incorporación laboral por parte de la empresa Auditel.

6.2. Trabajos Futuros

Como se acaba de comentar, con intención de seguir trabajando para las Salas de Teledocencia, existen varios objetivos que se desearía poder desarrollar, tanto para *Access Grid*, como para la aplicación web desarrollada por Miguel Ángel, que en su Proyecto de Final de Carrera hablará sobre ello.

Los futuros trabajos que se desean desarrollar son los siguientes:

Desarrollar nuevas Aplicaciones Compartidas

Tanto el personal de la Universidad de Cádiz, como el de Auditel, no conocían el desarrollo de Aplicaciones Compartidas para *Access Grid*. Cuando vieron en funcionamiento la aplicación *Shared PDF Viewer* quedaron muy sorprendidos y desean que se desarrollen nuevas aplicaciones. Por ejemplo, se finalizará de desarrollar la aplicación *Shared Evaluation*.

Una aplicación similar, también basándonos en el sistema *Adobe Connect*, es una aplicación para **diseñar encuestas** y que los usuarios puedan responderlas.

Otra aplicación que les gustarían que se desarrolle es un **reproductor multimedia compartido**, es decir, un reproductor multimedia, de forma que, al reproducir un vídeo o un sonido, el resto de participantes también lo reproduzcan.

Mejoras del sistema *Access Grid*

También se han propuesto varios objetivos para mejorar el sistema *Access Grid*, bien de forma directa, o bien, desarrollando aplicaciones de apoyo.

Uno de los objetivos primordiales es una mejor **Gestión de Salas Seguras**. La forma en que *Access Grid* gestiona el control de acceso a una Sala no es muy cómodo, ya que, hay que introducir "a mano", el *Distinguish Name* de cada usuario que se quiera permitir o denegar el acceso. Se pretende mejorar dicho sistema para automatizar más aún la gestión de control de acceso. Por ejemplo, se pretende que, en lugar de tener que introducir a mano cada usuario, aparezca un listado de usuarios conectados en una Sala de Bienvenida, donde todo el mundo tendría acceso. Posteriormente, el administrador del sistema *Access Grid*, con un simple arrastre, podrá asignarle un Rol o los permisos que considere oportunos para poder dar permiso o denegárselo.

Otra aplicación a desarrollar, en relación a ésta, es una herramienta de registro y/o invitación de usuarios. Esta aplicación se integraría en la aplicación web desarrollada por Miguel. Como el control de acceso se realiza a través de certificados digitales, esta herramienta, al registrar un usuario o invitarlo, se generaría un certificado digital para que se lo descargue y lo instale en su Cliente de *Access Grid*. Por otro lado, el sistema asignaría dicho certificado, automáticamente, a un rol determinado y, dependiendo de dicho rol, tendrá unos permisos u otros. Así, se ahorraría también el trabajo del administrador, ya que, el propio sistema daría o quitaría permisos al usuario registrado o invitado.

También se desea añadir las siguientes mejoras al sistema *Access Grid*:

- **Turnos de palabra:** Que un usuario pueda pedir permiso para hablar en la Sala.
- **Color por participante en el chat:** Actualmente todo está en modo texto simple, por lo que, cuando el chat tiene una gran cantidad de texto, se hace difícil seguir una conversación.
- **Control del tamaño de los ficheros a subir:** Actualmente *Access Grid* permite subir cualquier tipo de fichero y de cualquier tamaño, es importante restringir estas características para no saturar el Servidor.
- **Posicionamiento automático de ventanas:** Actualmente, la aplicación *Vic* tiene una opción de autoposicionar las ventanas, siguiendo un patrón (una ventana al lado de la otra o una ventana encima de la otra). Sin embargo, en algunos casos, interesa organizar las ventanas de una forma diferente, por ejemplo, colocar las ventanas de vídeo en la primera y en la tercera pantalla, dejando la pantalla central para una presentación. La idea es desarrollar una opción para que pueda almacenar este posicionado de ventanas y poder aplicarlo directamente sin tener que posicionarlas manualmente.
- **Poder expulsar un usuario no deseado:** Si se diera el caso de una intrusión o de un usuario que molesta, se desea añadir una opción para poder expulsar ese usuario de la Sala.
- **Enviar mensajes privados entre los usuarios:** Crear una especie de chat interno entre dos personas para enviarse mensajes privados que no quieren que otros lo lean.
- **Personalizar la interfaz:** Como un último objetivo, aunque es el de menos prioridad, es cambiar la interfaz principal de *Access Grid* para que sea mejor visualmente.

Aplicación web para reservar las Salas

También se desea desarrollar una aplicación web para **reservar las Salas**. Aunque actualmente las Salas no están muy solicitadas, principalmente, por un gran desconocimiento de su existencia, se prevee que, en no mucho tiempo, haya una mayor demanda de su uso. Ésto puede suponer un problema puesto que puede haber solapamiento a la hora de reservar las Salas.

Es por ello que surge la necesidad de crear una aplicación web donde los usuarios puedan reservar las Salas. Tendría disponible unos horarios a elegir, indicando que a esas horas las Salas están libres, así como el número de horas que necesita tenerla reservada, entre otros datos. Estas solicitudes lo recibiría algún personal de la Universidad de Cádiz y evaluará su solicitud en la misma aplicación, admitiendo o denegando (indicando el motivo) la solicitud.

Apéndices

Apéndice A

Documentación Generada - Manual de Access Grid

A.1. Breve introducción a AccessGrid

A.1.1. ¿Qué es *Access Grid*

Access Grid es un entorno integrado que soporta la comunicación de grupos utilizando redes de alta velocidad a través de Internet. *Access Grid* es un conjunto de recursos, incluyendo grandes pantallas multimedia, presentación y entornos interactivos y la compartición de aplicaciones en sus entornos de visualización. Estos recursos se utilizan para apoyar las interacciones entre grupos a través de la red. Ofrece audio de alta calidad y vídeo en tiempo real, que permite a grupos situados en varios lugares, interactuar y compartir datos e instrumentos científicos al mismo tiempo.

Figura A.1: Logo de Access Grid

Access Grid, además de servir para realizar reuniones entre individuos o grupos utilizando recursos de audio y vídeo, permite compartir aplicaciones simultáneamente entre los participantes, ya sea una presentación de *Powerpoint*, un documento en PDF, nuestro propio escritorio o un navegador web mientras estamos en Internet.

A.1.2. Una breve historia

Access Grid fue desarrollado por **FuturesLab** en el Laboratorio Nacional de Argonne, Chicago. Su primera aparición y primer evento a gran escala fue en el año 1999, en una serie de conferencias, llamada **"La Alliance Chautauqua 99"** que duró dos días, sobre la ciencia computacional organizada por la **NCSA**. *Access Grid* fue más tarde dado a conocer al público internacional en el evento **"Supercomputing '99 "** en Portland.

El primer nodo *Access Grid* de la costa oeste de Estados Unidos se instaló en el San Diego Supercomputer Center (UC San Diego) en enero de 2000 por B. Pailthorpe, J Moreland y N Bordes.

El 23 y 24 de marzo de 2000, se usó para albergar una reunión del West Coast / Washington DC President's Information Technology Advisory Committee (PITAC), donde participaron treinta CEOs de la costa oeste, quienes no tuvieron que desplazarse a Washington DC para tal reunión.

Posteriormente, *Access Grid* se extendió hacia Europa. *Access Grid* está soportado en la comunidad académica del Reino Unido, por el Centro de Soporte de *Access Grid* (Access Grid Support Centre, AGSC), fundado por JISC y gestionado por JANET.

El primer nodo *Access Grid* europeo fue construido en la Universidad de Manchester en 2001 y, posteriormente su AGSC comenzó en Abril de 2004.

Actualmente existen cerca de 300 nodos de *Access Grid* con AGSC en el Reino Unido, desde salas completamente equipadas hasta pequeñas salas de reuniones o despachos.

Existe un gran número de proyectos académicos usando la tecnología *Access Grid*, como los proyectos matemáticos Taught Course Centre y MAGIC.

A.1.3. Características de Access Grid

Access Grid, al tratarse de un sistema colaborativo, incluye una serie de características generales:

- Sistema de Multiconferencia con audio y vídeo, permitiendo establecer contacto, tanto visual, como auditivo, con varios participantes simultáneamente.

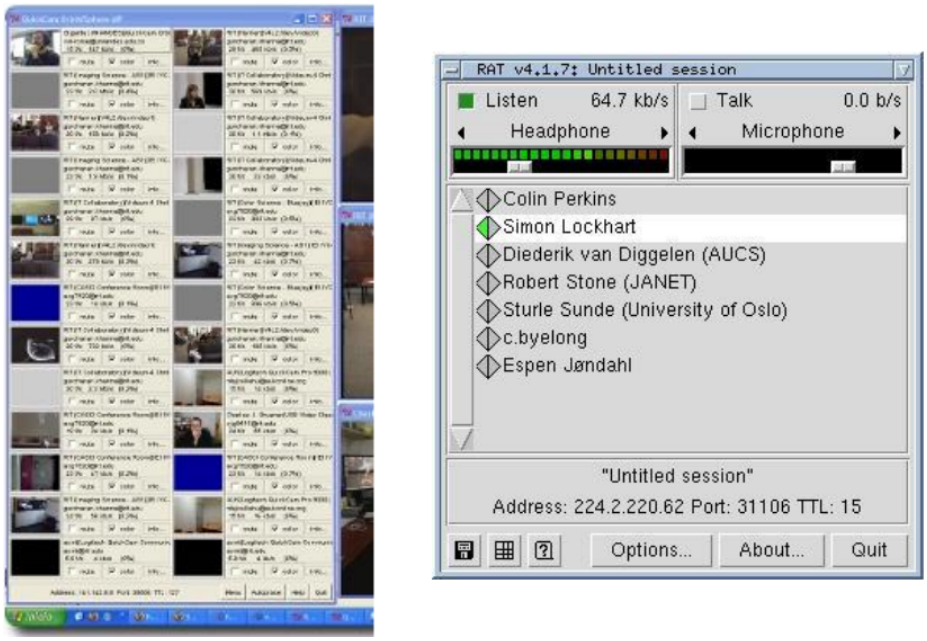


Figura A.2: Multiconferencia en Access Grid

- Capacidad para codificar en varios formatos y códecs, tanto en audio como en vídeo, tales como H.261, H.264, MPEG-4 e incluso formatos HD.
- Sistema de compartición de archivos entre los participantes.

- Sistema de chat entre los participantes.
- Conexiones multicast y unicast (a través de Puentes Multicasts), debido a la gran cantidad de información que viaja en una sesión de *Access Grid*, el sistema está ideado para trabajar bajo redes multicast, para no sobrecargar la red. En aquellos lugares donde no se disponga de Multicast, *Access Grid* permite utilizar Puentes Multicast, es decir, se conectaría al puente por Unicast, y dicho puente transmitiría toda la información por Multicast.
- Uso de aplicaciones compartidas. *Access Grid* integra varias aplicaciones tales como un navegador web compartido o una pizarra compartida, permitiendo, por un lado, que todos los participantes puedan interactuar en dicha aplicación como si fuera una sola y, por otro, transmitir cualquier cambio al resto de participantes al instante.
- Extensible. *Access Grid* dispone de una API para desarrolladores para corregir o mejorar el Sistema o para crear nuevas aplicaciones compartidas.

A.1.4. Arquitectura de Access Grid

Para comprender mejor el sistema Access Grid se explicará brevemente su arquitectura. El Sistema consta, básicamente, de dos componentes:

- **El Servidor de Salas:** Encargado de conectar a los clientes entre sí, habilitando un punto de encuentro común. Dentro de un mismo servidor de salas pueden existir tantas salas como sean necesarias. Dichas salas pueden ser públicas o privadas.
- **El Cliente de Sala:** Son los participantes en las reuniones. Un cliente puede ser desde un usuario con un pórtatil hasta una sala de conferencias completamente equipada con equipos de audio y vídeo de alta definición.

Los elementos principales que forman dicha arquitectura son:

- **VenueServer:** Servidor de Salas
- **Venue:** Una sala
- **VenueClient:** Cliente de Sala
- **Node:** Un Nodo

La siguiente figura muestra cómo se conectan estos elementos formando la estructura de Access Grid:

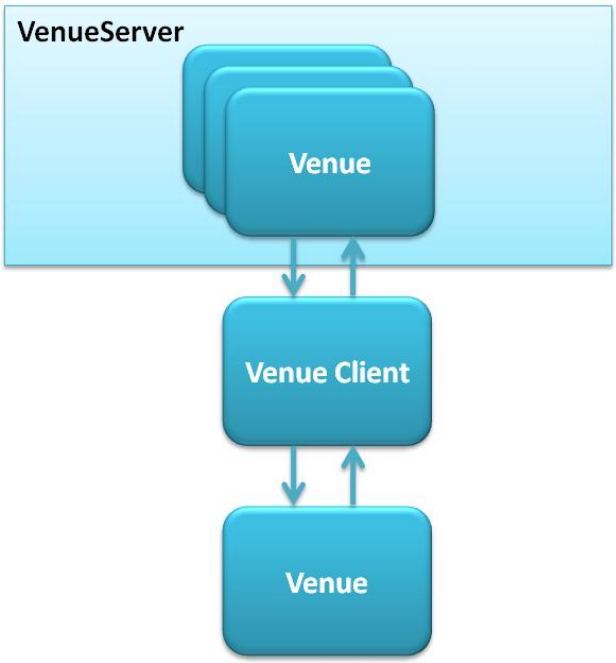


Figura A.3: Arquitectura de Access Grid

El Servidor de Salas (**VenueServer**) gestiona un número de Salas (**Venues**), al igual que un servidor web gestiona páginas web. El Cliente de Sala (**VenueClient**), por un lado se conecta a una Sala y luego se comunica con el Nodo para realizar una determinada acción en dicha Sala; por ejemplo, iniciar el servicio de audio para la Sala en la que se encuentra.

Arquitectura del Nodo

El elemento Nodo contiene una colección de componentes software ejecutándose en un Cliente de Sala para controlar herramientas que reciben y emiten audio y vídeo, entre otras cosas.

El Cliente de Sala comunica al Servicio de Nodo (**NodeService**) de cambios de estado. El Servicio de Nodo agrega los llamados **ServicesManagers**, que son los que se encargan de gestionar los Servicios en las máquinas locales, y recoge información sobre los **Servicios** que se ejecutan en ellas pudiéndose comunicar con los Servicios cuando sea necesario.

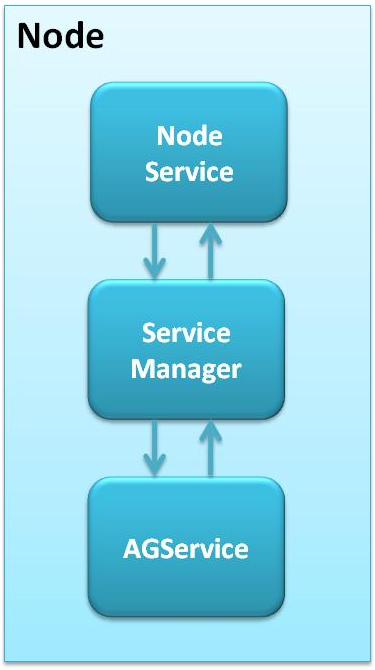


Figura A.4: Arquitectura de un Nodo Access Grid

Así, pues, vemos que los componentes que forman la arquitectura del Nodo son:

- **NodeService:** Servicio de Nodo. Agrega ServicesManagers de máquinas locales, además de guardar configuraciones en las que se incluyen los servicios, sus descripciones y sus configuraciones individuales.
- **ServiceManager:** Gestor de Servicios. Se encarga de localizar los recursos disponibles en la propia máquina, y en máquinas locales. Además, se encarga de gestionar los servicios.
- **Services:** Los propios Servicios. Lee y escribe configuraciones, como por ejemplo el número y el tipo de la tarjeta de vídeo, el códec de vídeo a usar, la calidad, formato de pantalla, formato del audio, calidad de audio, etc., además, de describir las características de los propios servicios. Por último, ejecuta software subyacente (como *vic* para los servicios de vídeo o *RAT* para los servicios de audio).

Todos los elementos pueden ser llamados a través de la interfaz **SOAP**.

Flujo de Datos al acceder a una Sala

Cuando el Cliente de Sala entra en una Sala se realizan una serie de operaciones para establecer las direcciones de los medios conectados:

- Consulta al Servicio de Nodo las Funcionalidades que ofrecen sus Servicios.
- Transporta dichas Funcionalidades a la Sala (fase de Negociación de Funcionalidades, en inglés **NegotiateCapabilities**).
- La Sala (en la etapa de Negociación de Funcionalidades) asocia las Funcionalidades de entrada con las Funcionalidades de los medios de comunicación conectados que ya han sido establecidos. Si no se encuentran ajustes, la Sala asigna una nueva dirección al medio de comunicación. El

conjunto de direcciones de los medios de comunicación es devuelto al Cliente de Sala (como **StreamDescriptions**).

- El Cliente de Sala pasa estos StreamDescriptions al Servicio de Nodo, que los coloca en servicios individuales basándose en dichas Funcionalidades. Por ejemplo, el stream de audio se pasa al Servicio de Audio, porque tiene una Funcionalidad de tipo *"audio"*.

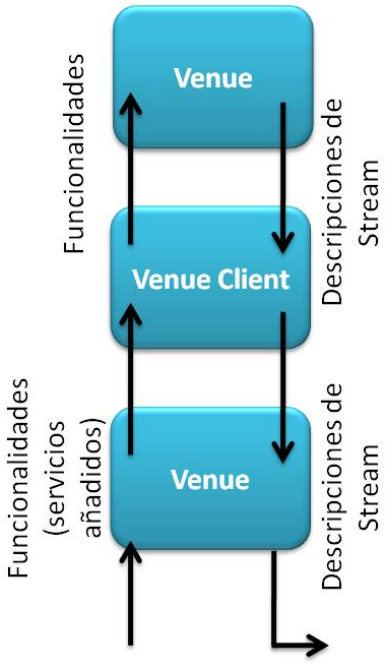


Figura A.5: Flujo de Datos al acceder a una Sala

A.2. Descarga e Instalación del Sistema

La descarga e instalación del sistema Access Grid comprende la instalación tanto del Servidor como del Cliente, es decir, cuando se descargue e instale Access Grid se instalarán tanto las aplicaciones necesarias para ser un Servidor, como las aplicaciones necesarias para ser un Cliente. Dependiendo de la funcionalidad que se le vaya a dar a un determinado PC se utilizarán y configurarán las correspondientes aplicaciones.

A.2.1. Instalación en Ubuntu

Nota: En los pasos detallados se utilizará como ejemplo la distribución Ubuntu Karmic Koala

- Lo primero es instalar la llave pública del repositorio de UQVislab. Para ello introducimos estos comandos desde consola (Menú Aplicaciones $->$ Accesorios $->$ Terminal):

```
wget http://www.vislab.uq.edu.au/debuntu/uqvislab-pubkey.asc
sudo apt-key add uqvislab-pubkey.asc
```

- Seguidamente hay que añadir los repositorios de UQVislab, así que también desde consola:

```
sudo wget http://www.vislab.uq.edu.au/debuntu/sources.list.d/karmic.list
    -O /etc/apt/sources.list.d/uqvislab.list
```

- Actualizamos la lista de paquetes e instalamos AccessGrid3.2 (es una versión beta, la última versión estable es la 3.1, aún así los desarrolladores recomiendan instalar esta versión):

```
    sudo apt-get update
    sudo apt-get install accessgrid3.2
```

- Ya sólo queda ejecutar el cliente desde "Aplicaciones $->$ Access Grid 3 $->$ AG Venue Client" de nuestro menú Gnome.

A.2.2. Instalación en Windows

Para descargar el sistema Access Grid:

- Diríjase a su página oficial [1] y pulse en el apartado **Software**:

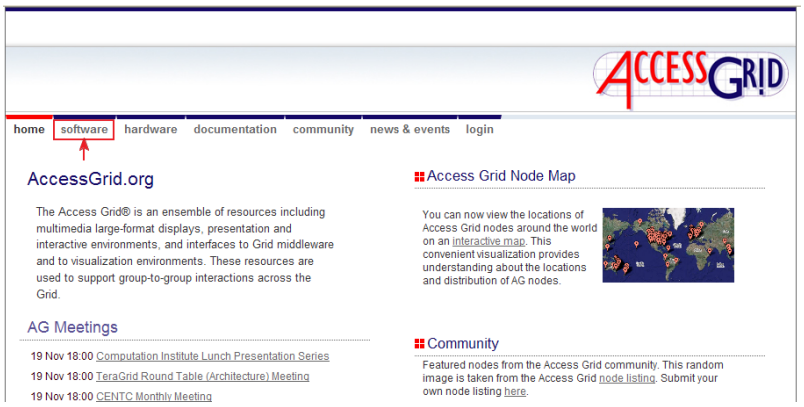


Figura A.6: Página Oficial de Access Grid

- En la siguiente ventana, le aparecerá en la derecha un menú donde aparecen las últimas versiones disponibles para varios Sistemas Operativos. En nuestro caso, nos dirigiremos al apartado **Windows**. Vemos que hay dos descargas:

  - **Access Grid (*installer only*)**: Para descargar el sistema Access Grid únicamente, sin las dependencias necesarias (recomendado).
  - **Access Grid Bundle (*includes required dependencies*)**: Para descargar el sistema Access Grid junto a las dependencias necesarias.

[1] http://www.accessgrid.org/

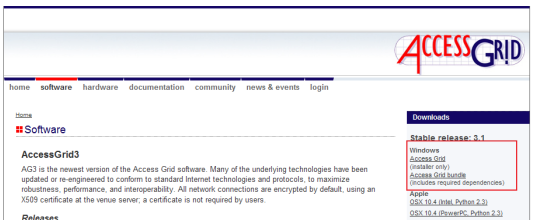


Figura A.7: Instaladores de Access Grid para Windows

- Descargamos la primera opción y, una vez descargado, lo ejecutamos. Nos aparecerá la pantalla de bienvenida. Pulsamos ”Siguiente” y nos aparecerá la siguiente pantalla:

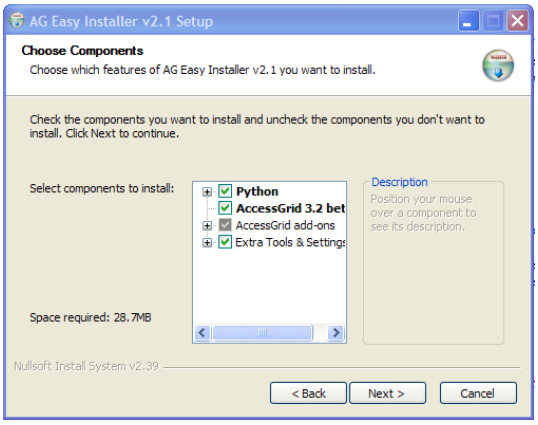


Figura A.8: Instalador de Access Grid para Windows

- En dicha pantalla nos solicita qué componentes queremos instalar. Como se ha comentado anteriormente, si no dispone de ninguna de las dependencias necesarias, déjelo todo activado y pulse Siguiente. En caso de tener alguna dependencia ya instalada, desmárquela y pulse Siguiente.

- Por último, se nos solicitará la ruta donde queremos instalar el sistema Access Grid. La ruta por defecto es `"C:\Archivos de programa\AGTk-3"`.

- Pulsamos ”Instalar” y comenzará la instalación.

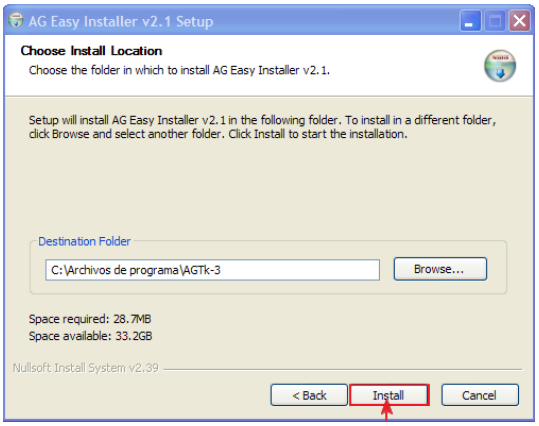


Figura A.9: Finalizando la instalación de Access Grid

- Una vez finalizado el proceso de instalación, nos aparecerá la pantalla de finalización. Pulsamos el botón ”Finish” para terminar.

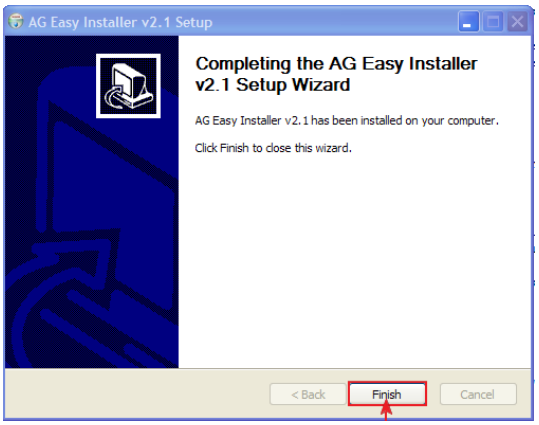


Figura A.10: Instalación finalizada de Access Grid

A.2.3. Instalación del Software Adicional

El instalador de Access Grid, además de incluir todas las dependencias necesarias para que el sistema Access Grid funcione, se incluye, además, software y herramientas adicionales. Basta con ejecutar el instalador del sistema y, después de la pantalla de bienvenida y pulsar ”Siguiente”, observamos la siguiente pantalla:

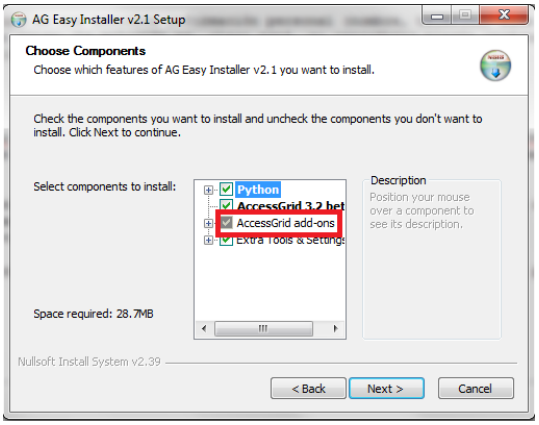


Figura A.11: Instalador de Access Grid

Vemos, observando el recuadro rojo, que, además de instalar Access Grid y sus dependencias, incluye una sección aparte para instalar software adicional. Desplegamos esta sección y nos aparece lo siguiente:

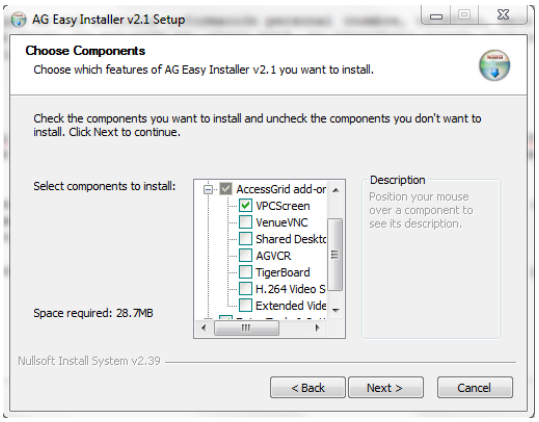


Figura A.12: Software adicional de Access Grid

El software adicional que se incluye en el instalador es el siguiente:

- **VPCScreen:** Esta herramienta es un servicio añadido a Access Grid, en el que nos permite emitir nuestro escritorio a los demás clientes conectados a la sala. Se puede configurar para emitir todo el escritorio, una porción de ventana, o incluso una aplicación en concreto.
- **VenueVNC:** Esta aplicación es la ya conocida VNC, utilizada para acceder, a través de escritorio remoto, a un PC determinado.
- **Shared Desktop:** Esta aplicación, como su nombre en inglés indica, permite al usuario compartir su escritorio al resto de clientes conectados en el sistema *Access Grid*.
- **AGVCR:** Esta aplicación nos permite grabar a un fichero una sesión de *Access Grid*.
- **TigerBoard:** Esta herramienta es un servicio añadido a *Access Grid*, en el que nos proporciona una pizarra electrónica con la que podremos hacer uso y el resto de clientes conectados a la sala de Access Grid, podrán visualizar su contenido.

- **H.264 Video Services:** Esta herramienta nos instala unos servicios de vídeo añadidos a Access Grid. Estos servicios se diferencian de los habituales en el uso de un códec de mayor calidad. Los servicios de vídeo que vienen instalados por defecto junto a *Access Grid* utilizan el códec H.261. Dicho códec ofrece muy poca calidad de vídeo y a una baja resolución de $320x240$, por lo que, para el uso de las aulas, donde hay instaladas pantallas de grandes dimensiones, puede que dicho códec sea insuficiente al ofrecer una baja calidad. Este paquete instala los mismos servicios de vídeo, pero utilizando permitiendo utilizar los códecs H.264 ó MPEG-4, ofreciendo mayor calidad de vídeo (incluso con un bajo bitrate) y una mayor resolución de pantalla.
- **Extended Video Services:** Al igual que el paquete anterior, este paquete también instala servicios de vídeo añadidos a *Access Grid*. A diferencia del anterior, éste nos permite utilizar códecs para emitir y recibir vídeo en calidad HD, ofreciendo aún más calidad de vídeo y mayor resolución que el anterior (a costa, claro está, de un mayor consumo de ancho de banda).

A.3. El Sistema Access Grid

En este capítulo se explicará con detalle cómo utilizar y configurar el Sistema Access Grid.

Principalmente, Access Grid se divide en dos grandes grupos:

- **El Servidor de Salas (*VenueServer*)**: El Servidor de Salas es el software que ofrece una interfaz para configurar nuestro Servidor de Access Grid. En él se pueden crear tantas Salas como las que sean necesarias, además de otras opciones como la encriptación del contenido multimedia, creación de roles y asignarles permisos.
- **El Cliente de Sala (*VenueClient*)**: El Cliente de Sala es el software encargado de conectarse y participar en una Sala Virtual de Access Grid. En él, se visualizan los participantes conectados actualmente en una Sala concreta, otras conexiones para acceder a otras Salas, los contenidos que dispone la Sala, además de una interfaz para su configuración.

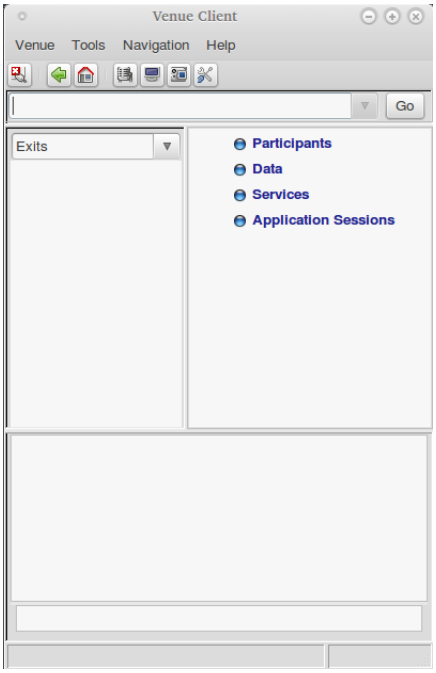


Figura A.13: Cliente Access Grid

A.4. El Servidor de Salas

Como se ha comentado, el Servidor de Salas es el software que ofrece una interfaz para configurar nuestro Servidor de Access Grid. En él se puede crear tantas Salas como las que sean necesarias, además de otras opciones como la encriptación del contenido multimedia, creación de roles y asignarles permisos.

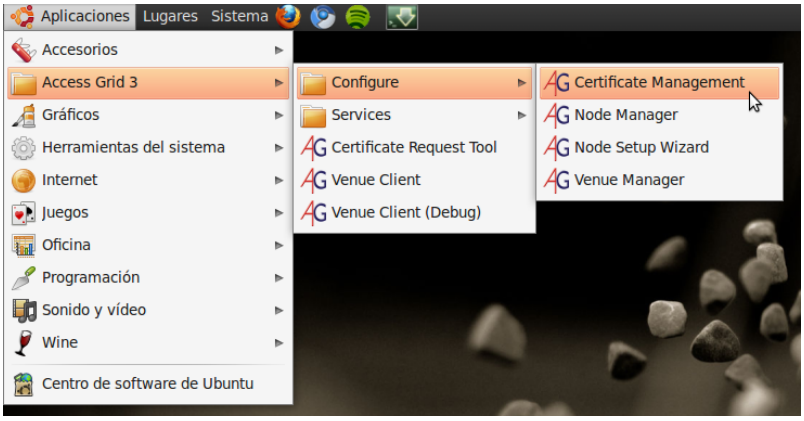
Todo el sistema de Access Grid, de cara a la seguridad, funciona bajo **certificados digitales**, garantizando una buena seguridad.

Así pues, para crear nuestro Servidor de Salas, lo primero que se necesita es crear un certificado digital. Access Grid incluye una herramienta para solicitar un certificado digital para nuestro Servidor de Salas y, una vez lo firmen, podremos instalarlo.

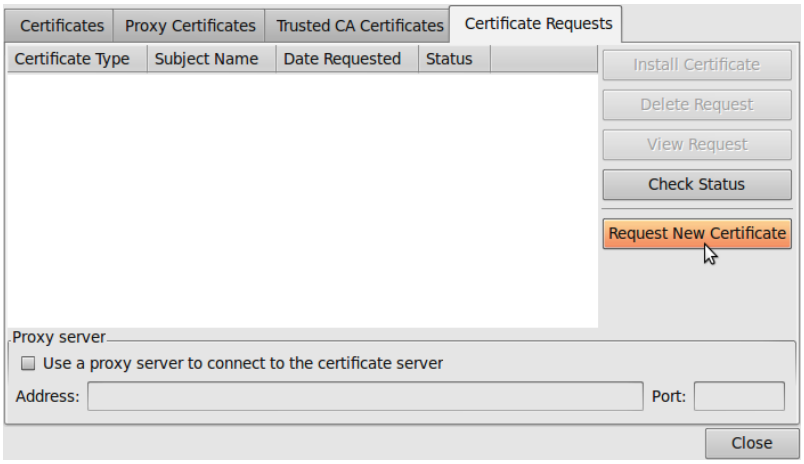
A continuación se detallan los pasos a seguir:

A.4.1. Solicitud e Instalación del Certificado

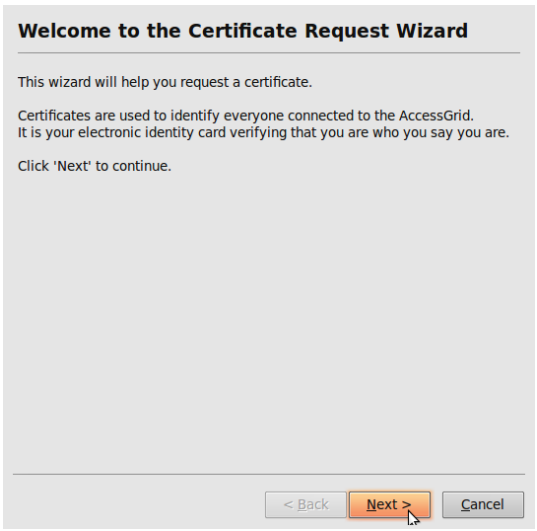
- Vamos a "Aplicaciones $->$ Access Grid 3 $->$ Configure ->Certificate Management":



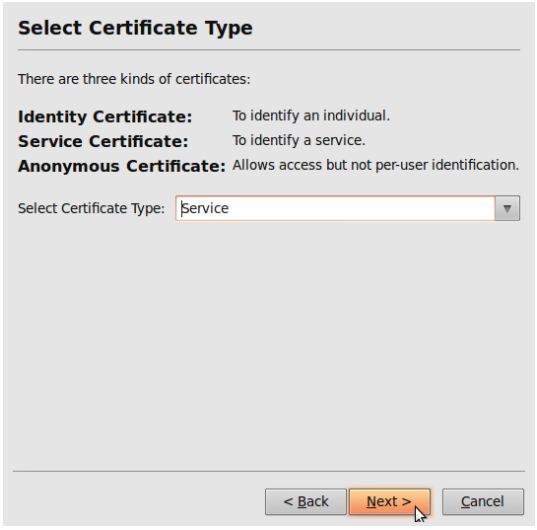
- En la ventana que se nos abre, vamos a la pestaña ”Certificate Requests” y allí hacemos click sobre el botón **Request New Certificate**:



- Aparece el Asistente de petición del certificado. Pulsamos en ”Next >” para avanzar:



- En ”Select Certificate Type” seleccionamos **Service** y pulsamos sobre ”Next >”:

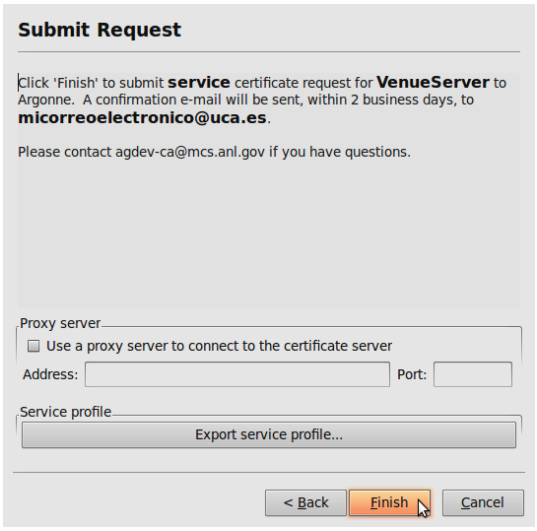


- Ahora en ”Service Type” seleccionamos **Venue Server**. En ”Machine Name” ponemos el nombre de nuestra máquina (en Ubuntu puede verse y modificarse en `/etc./hostname`) seguido de nuestro dominio (en nuestro caso *uca.es*).

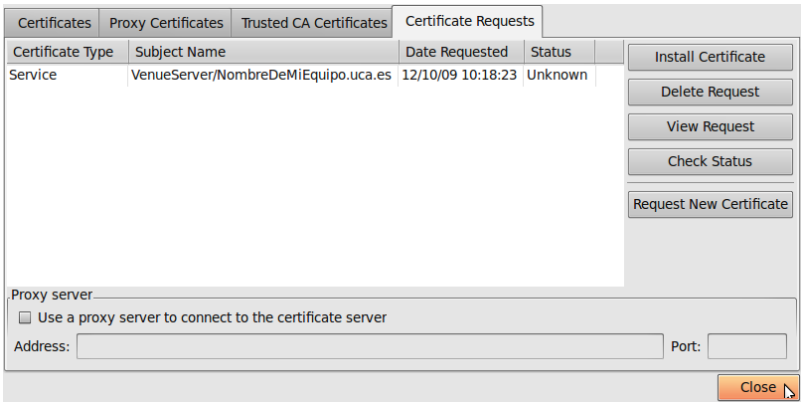
  En ”E-mail” introducimos una dirección de correo electrónico válida que será donde nos confirmen su aceptación. [2] Pulsamos sobre ”Next >” para finalizar.

- Nos aparece una nueva ventana donde nos dice que nos llegará un correo electrónico de confirmación antes de 2 días laborables. Pulsamos sobre **Finish**:

[2] Atención: la dirección de correo electrónico debe pertenecer al mismo dominio de la máquina.



- Ahora vemos en la ventana del *Certificate Manager* que tenemos pendiente la aprobación del certificado. Pulsamos sobre ”Close”.



- Cuando hayan validado nuestro certificado (recibiremos un email de confirmación), pulsamos el botón Check Status y podremos ver que nuestro certificado ya ha sido firmado. Pulsamos el botón **Install Certificate** y el certificado se instalará en nuestro Servidor de Salas. A partir de ahora podremos utilizarlo.

A.4.2. El Gestor de Salas (*VenueManager*)

Una de las acciones que deseemos realizar en nuestro Servidor de Salas es el poder crear, configurar/editar y eliminar Salas en él. Para ello, el sistema Access Grid incluye una herramienta muy útil, llamada **VenueManager**, la cual, principalmente, nos permite realizar este tipo de acciones.

Dicha aplicación se encuentra en ”Aplicaciones $->$ Access Grid 3 $->$ Configure $->$ Venue Manager”, o bien, desde una terminal, podemos ejecutarlo escribiendo ***VenueManagement3.py***.

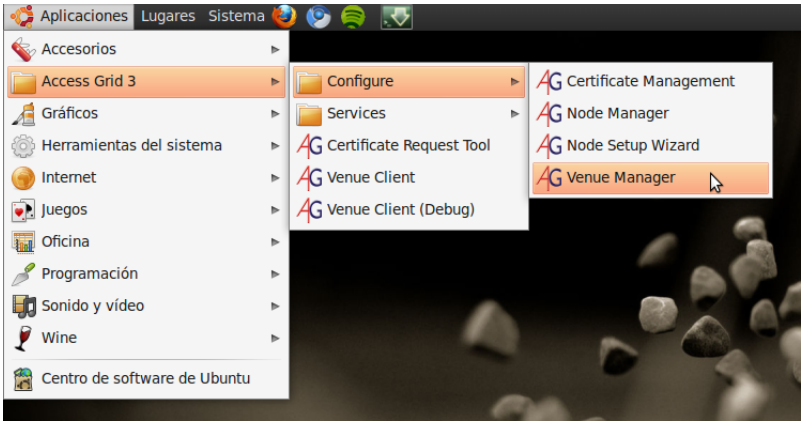


Figura A.14: Ejecutando el Gestor de Salas

Para configurar nuestro Servidor de Salas, en dicha aplicación debemos de introducir la dirección de nuestro Servidor. Para ello, o bien lo hacemos desde la máquina donde se ejecute el Servidor de Salas, o bien, debemos de tener permisos para poder acceder remotamente (en próximas secciones se explicará sobre permisos).

Una vez introducida la dirección, pulsamos sobre **Go** y accederemos a nuestro Servidor de Salas. A partir de aquí, podremos configurarlo a nuestro gusto.

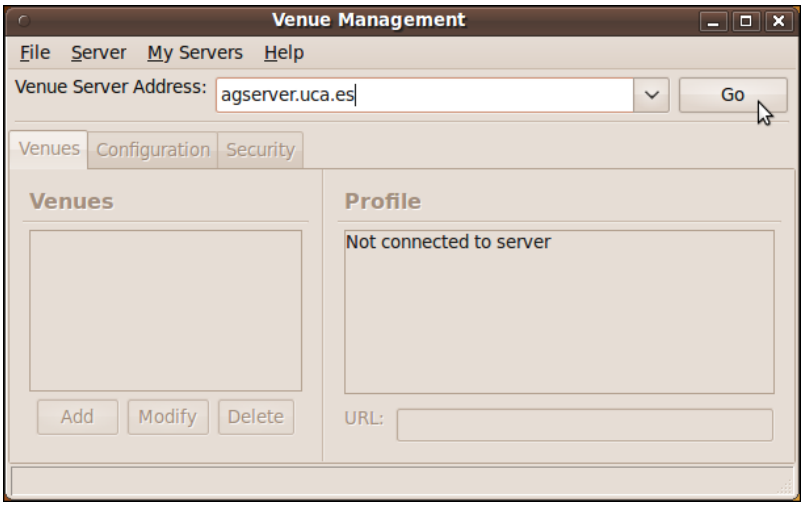


Figura A.15: Ventana principal del Gestor de Salas

Si lo desea, para no volver a introducir la dirección completa del Servidor de Salas, puede agregarlo a una lista, parecido a los favoritos de los navegadores de internet. Para ello:

- Diríjase a MyServers $->$ Add Server:

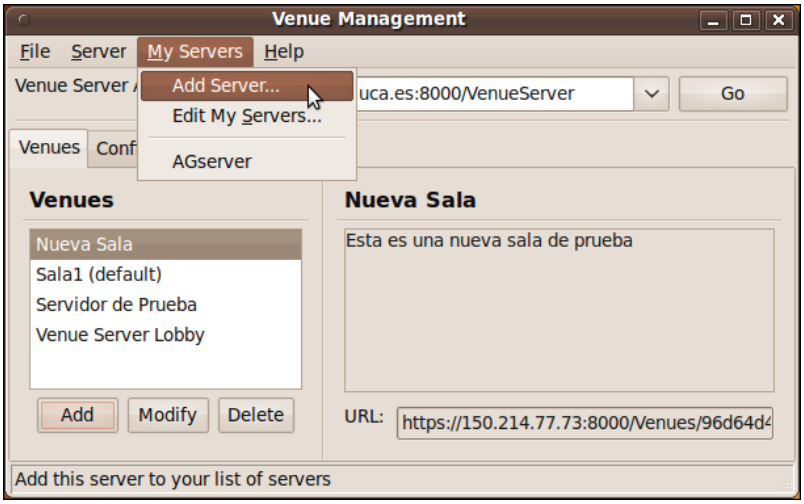


Figura A.16: Añadir un Servidor de Salas a la lista

- Introduzca en **Name** el nombre que desea darle al Servidor de Salas, y en **URL** introduzca la dirección de dicho Servidor de Salas. Por último, pulse OK.

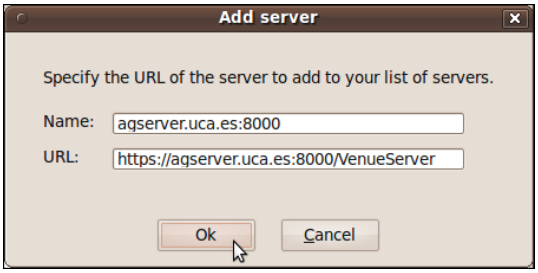
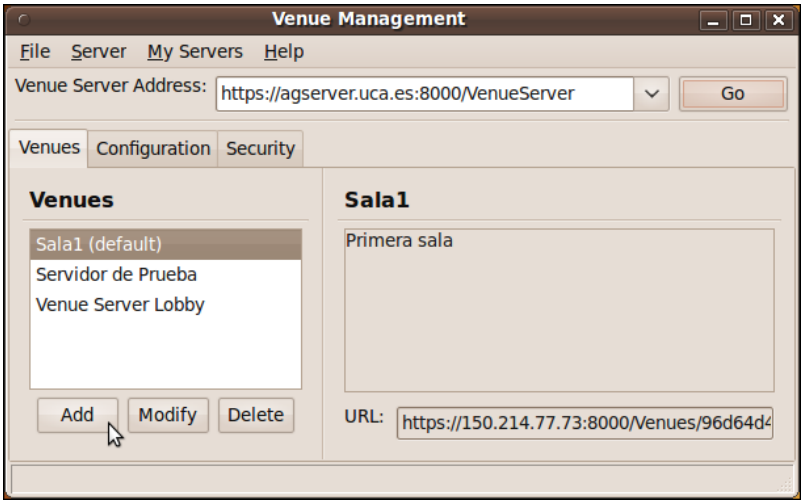


Figura A.17: Introduciendo los datos del Servidor de Salas

- Ahora le aparecerá lo que haya introducido en **Name** en un listado dentro de *MyServers*.

Configuración General del Servidor de Salas

Una vez introducida la dirección del Servidor de Salas y pulsar el botón **Go** nos deberá de aparecer lo siguiente, indicándonos que hemos entrado correctamente en el Servidor de Salas:

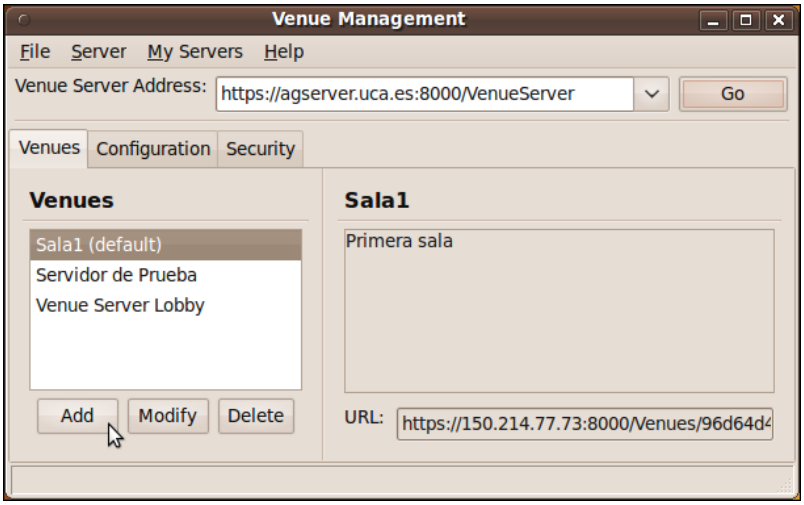


Una vez dentro vemos que la ventana del Gestor de Salas se divide en tres pestañas:

- **Venues:** Conjunto de opciones relacionadas con la gestión de Salas
- **Configuration:** Conjunto de opciones relacionadas con el direccionamiento y la encriptación
- **Security:** Conjunto de opciones relacionadas con el control de acceso al Servidor de Salas

Venues (Salas)

En esta pestaña podremos gestionar las Salas que dispondrá nuestro Servidor de Salas. En ella podremos crear Salas, editarlas o borrarlas.

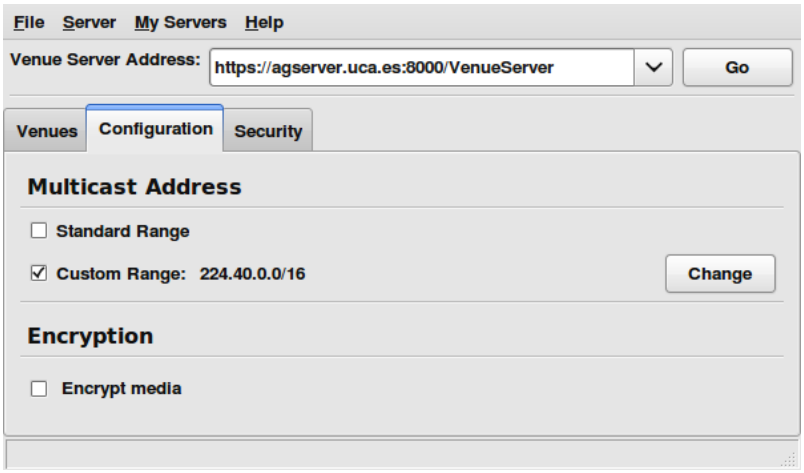


Podemos ver en dicha ventana la siguiente información:

- A la izquierda aparece el listado de Salas, dentro del bloque Venues.
- A la derecha aparece la descripción asociada a la Sala seleccionada junto a la URL que se le ha asignado (generado)
- Por último aparecen los siguientes botones en la parte inferior izquierda:

- **Add:** Para añadir/crear una nueva Sala
- **Edit:** Para editar/configurar una Sala ya creada
- **Delete:** Para eliminar la Sala seleccionada

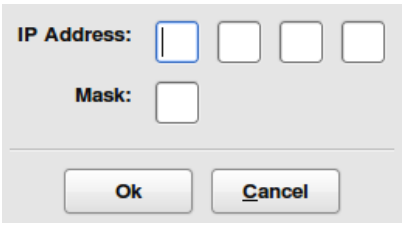
Configuration



Desde esta pestaña podemos configurar el direccionamiento para la red multicast y la encriptación de los contenidos multimedia que viajen en una sesión de Access Grid.

Multicast Address

- **Standard Range:** Si marcamos esta opción, Access Grid asignará dinámicamente una dirección IP dentro del rango multicast. Por cuestiones de seguridad, se recomienda utilizar esta opción, a menos que, por otros motivos, esté obligado a introducir una dirección estática. Para ello marque la opción siguiente.
- **Custom Range:** Si marcamos esta opción podremos introducir una dirección IP estática (fija), dentro del rango multicast. Para ello, pulsamos el botón Change y nos aparecerá la siguiente ventana:



- En **IP Address** introducimos los valores de la IP que queremos establecer
- En **Mask** introducimos el valor del puerto que queremos establecer

Encryption

Encrypt media: Si activamos esta opción Access Grid encriptará todo flujo de información que viaje por él, añadiendo seguridad para evitar intrusiones o acceso a contenido no deseado. Es decir, si

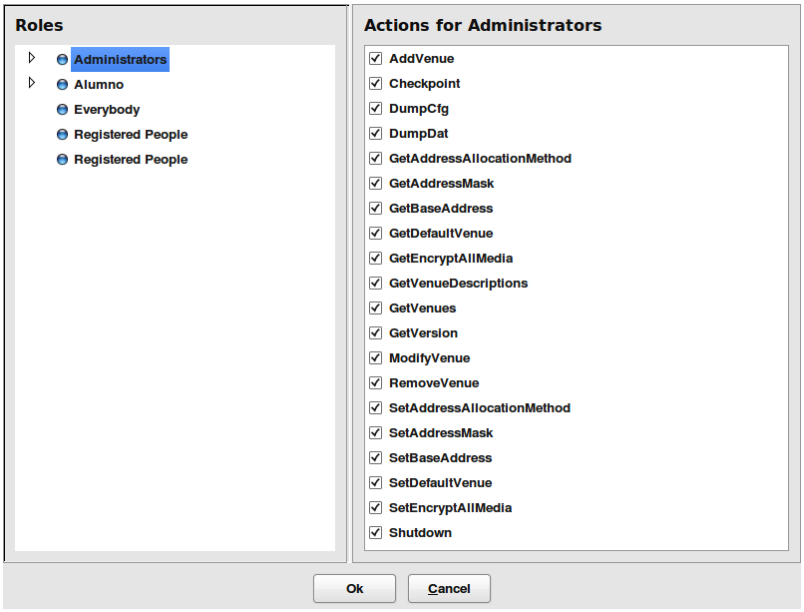
no activamos esta opción, una persona puede captar un flujo si conoce la IP de dicho flujo, por ejemplo, si en una sesión de Access Grid donde los participantes emiten vídeo, si la IP del vídeo de uno de los participantes es el 224.41.101.101:8000, y tenemos esta opción desactivada, otra persona puede capturar su vídeo emitido desde una aplicación determinada introduciendo esta IP. Lo mismo ocurre con el audio, por lo que, normalmente, no es deseado que esto ocurra. Así que lo recomendado es activar esta opción para añadir algo de seguridad a nuestro Servidor de Salas.

Security

En esta pestaña podemos gestionar el acceso al Servidor de Salas, permitiendo/denegando el acceso a personas para configurar el propio Servidor de Salas. Es decir, por defecto, para configurar el Servidor de Salas hay que ejecutar la aplicación Venue Management en la propia máquina del Servidor de Salas. Sin embargo, si gestionamos los permisos, podemos darnos permisos, a través de los **certificados digitales**, para poder configurar el Servidor de Salas desde otro PC.

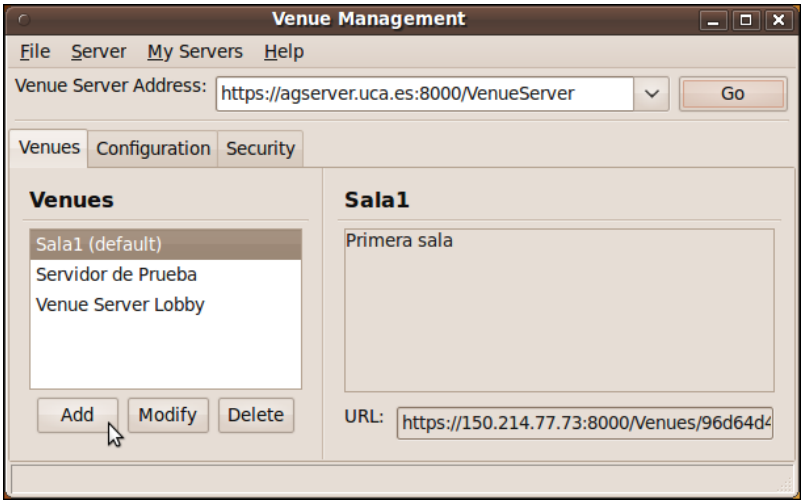
Para ello existen un conjunto de **Roles, Participantes y Acciones** (permisos). Podemos crear/eliminar Roles y crear/eliminar Participantes. Por cada Rol podemos activar/desactivar Acciones. Por último, podemos asignar a los participantes uno o varios Roles.

Todo esto se realiza pulsando el botón **Manage Security** que aparece la siguiente ventana:

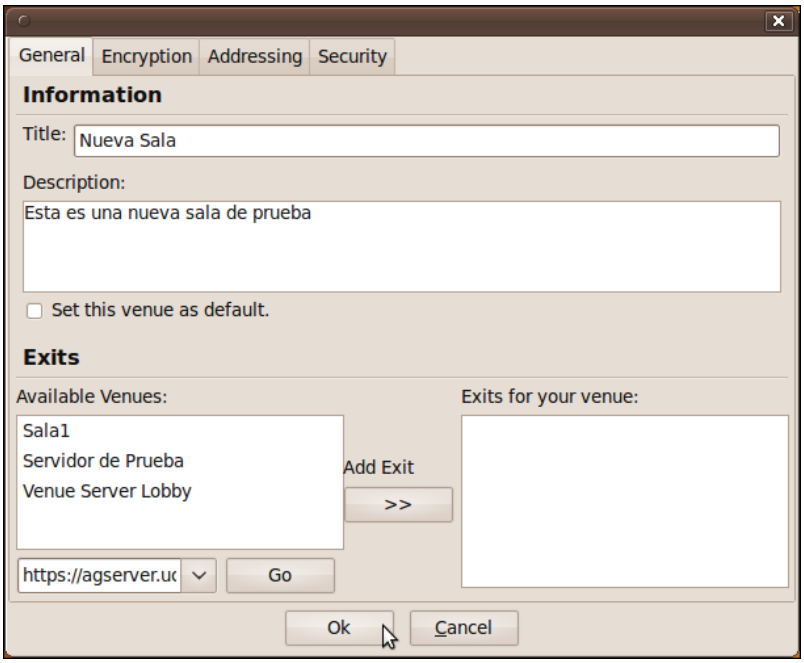


A.4.3. Cómo crear Salas

Una vez introducida la dirección del Servidor de Salas y pulsar el botón Go nos deberá de aparecer lo siguiente, indicándonos que hemos entrado correctamente en el Servidor de Salas:

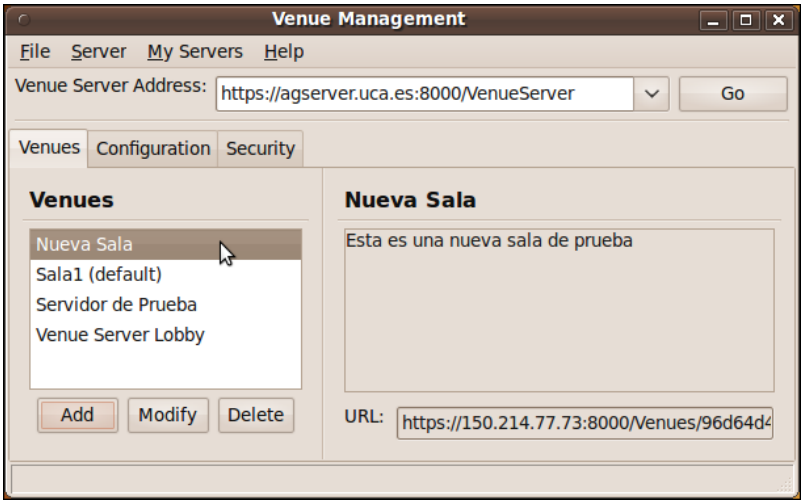


Una vez dentro pulsamos el botón **Add** y nos aparecerá la siguiente ventana:



En la nueva ventana introducimos el nombre de la sala en ”Title” y una descripción en ”Description”. Si marcamos la casilla **Set this venue as default**, esta sala será donde entren por defecto los usuarios al conectarse al servidor. Pulsamos sobre ”Ok” para terminar de añadirla.

Cuando volvamos a la ventana principal del Gestor de Salas veremos nuestra nueva sala creada:

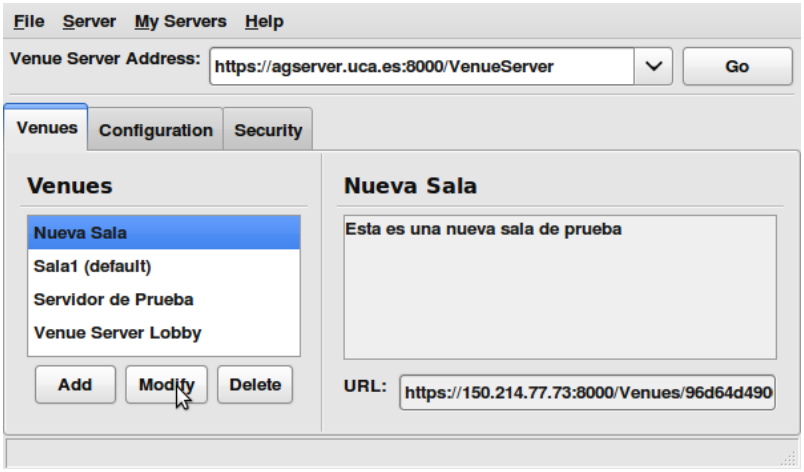


A.4.4. Editar una Sala

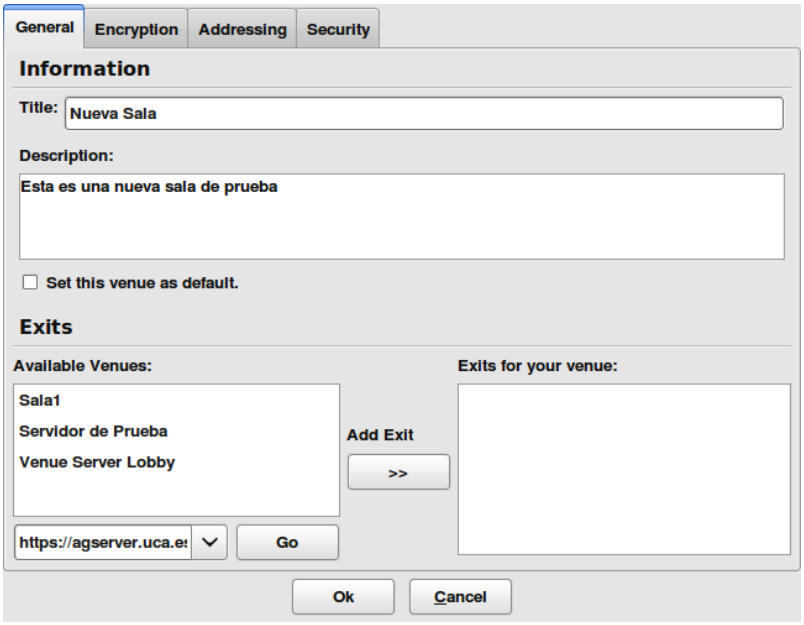
Como es habitual en cualquier sistema o software cuando creamos algo, podemos editarlo, bien para corregir algún dato que hayamos introducido por error, o bien para añadir o quitar funcionalidades a dicha creación.

En nuestro caso, podemos editar una Sala para corregir algún dato introducido erróneamente. Además, una vez que creamos una Sala, normalmente vamos a necesitar editarla, ya que, a la hora de crear la Sala no nos permite realizar algunas acciones que sí necesitaremos, como, por ejemplo, el control de acceso a la Sala.

Para editar una Sala, dentro del Gestor de Salas (*Venue Management*), y, una vez introducida la dirección del Servidor de Salas, seleccionamos la Sala que queremos editar y pulsamos el botón **Edit**:



Nos aparecerá la siguiente ventana (puede tardar un poco en aparecer debido a las conexiones y a la concurrencia):

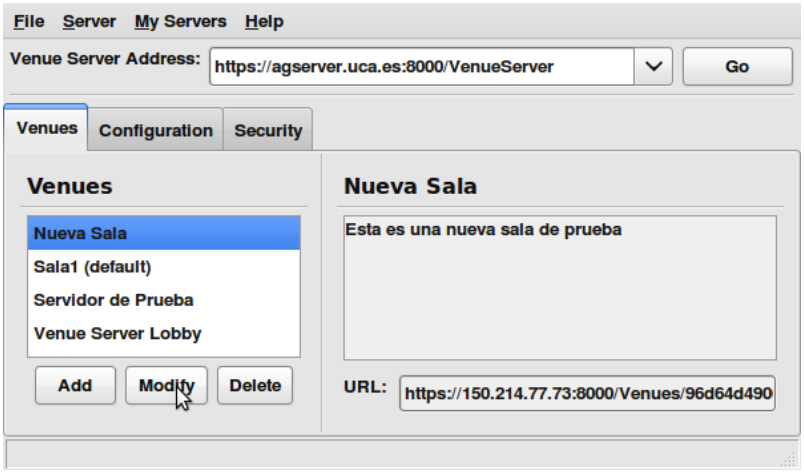


Podemos observar que dicha ventana está dividida en cuatro pestañas:

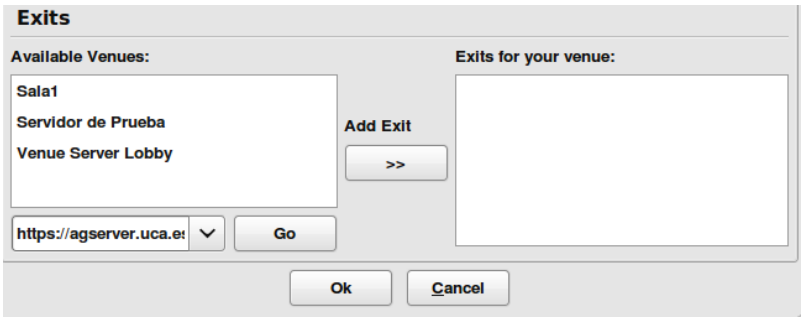
- **General:** Conjunto de opciones generales de una Sala (su nombre, descripción, etc.)
- **Encryption:** Conjunto de opciones relacionadas con la encriptación del flujo multimedia de la Sala seleccionada a editar
- **Addressing:** Conjunto de opciones relacionadas con el direccionamiento de los flujos de audio y vídeo
- **Security:** Conjunto de opciones relacionadas con el control de acceso a la Sala seleccionada

A continuación se detallan cada una de ellas:

General



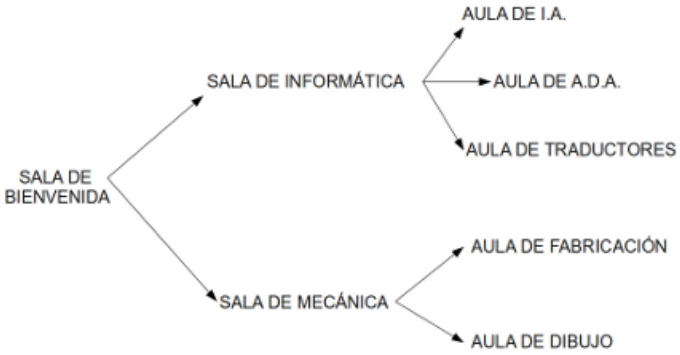
Esta ventana le parecerá familiar ya que es la misma que usó a la hora de crear una Sala. Aquí podrá editar el nombre de la Sala, editando el texto introducido en **Title**; y su descripción, editando el texto introducido en **Description**; además de poder activar o desactivar que la Sala sea la Sala por defecto, activando o desactivando casilla **Set this venue as default**.



Sin embargo, hay una opción que no se explicó en la creación de Salas, y puede serle muy útil. De hecho puede realizar estos pasos a la hora de crear una Sala. Se trata del grupo de opciones que está dentro de **Exits**, es decir, las Salidas.

Las Salidas son otras Salas en las que, desde la Sala donde nos encontremos, podremos ”viajar” hacia ellas. La finalidad principal de esta utilidad es la de facilitar al usuario la disposición de las Salas que él necesita o le interesa acceder. Es decir, en un Servidor de Salas puede haber 100 Salas, pero sin embargo, al grupo de participantes de la carrera de Electrónica, únicamente necesite sus correspondientes Salas de Electrónica, por lo que les sería muy laborioso buscar en las Salas que necesitan entre las 100 que hay.

Otro motivo es el **jerarquizar las Salas**, en el sentido de que en una Sala se descompone en otras Salas. Por ejemplo, una posible jerarquización sería la siguiente:



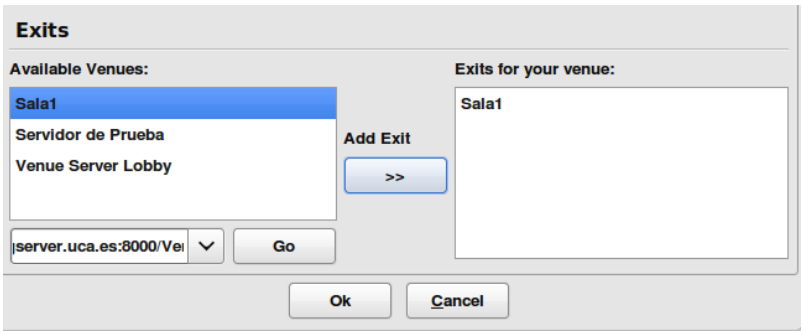
Por lo que esta herramienta sería necesaria utilizarla.

Para indicar las Salidas que tiene una Sala hacemos lo siguiente:

- En el lado izquierdo, dentro del grupo **Available Venues**, aparecen todas las Salas disponibles en un Servidor de Salas. Justo abajo de este bloque hay una entrada de texto para introducir una dirección diferente de Servidor de Salas, pulsando sobre Go, accederá al Servidor de Salas y se

listarán en Available Venues las Salas que dispone. Es decir, una Sala puede tener Salidas (otras Salas) que estén bien en el mismo Servidor de Salas, o bien, que estén alojados en otro Servidor de Salas.

- Una vez encontremos la Sala que deseemos asignarle como Salida, lo seleccionamos y pulsamos el botón **Add Exit** (Añadir Salida). La Sala seleccionada aparecerá dentro del grupo **Exits for your venue** (Salidas para su Sala):



Editar las Salidas

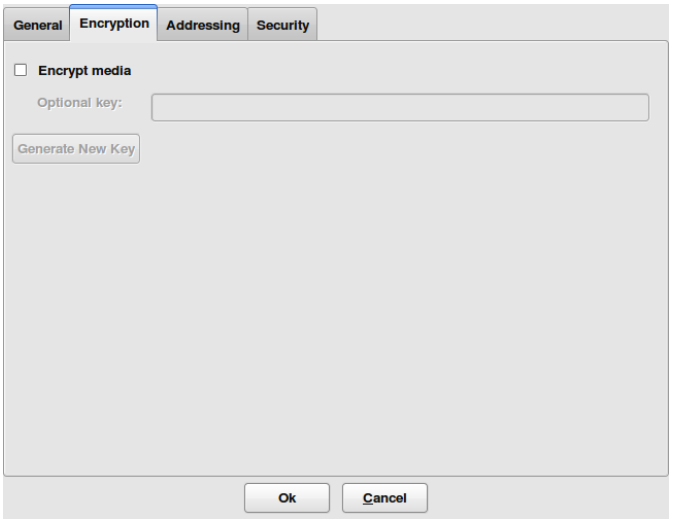
Si tenemos en una Sala asignado un conjunto de Salas como Salida, dentro del grupo *Exits for your venue*, si pulsamos el segundo botón del ratón, nos aparecen las siguientes opciones:

- **Add Exit...**: Nos permite añadir otra Sala como Salida, introduciendo un nombre y su URL. Recalcar que hay que introducir la URL de la Sala que Access Grid generó a la hora de crearla, no la URL del Servidor de Salas. Recuerde que, para saber qué URL tiene una Sala, desde el Gestor de Salas, una vez introducida la dirección del Servidor de Salas, seleccionando una Sala de las que hay disponible aparece en el cuadro URL su dirección generada.

- **Edit...:** Nos permite editar la Salida seleccionada, cambiando bien su nombre (si queremos darle otro nombre más identificativo), o bien, su URL (si, por error hemos introducido una URL errónea).

- **Remove:** Elimina la Sala de la lista de Salidas

Encryption

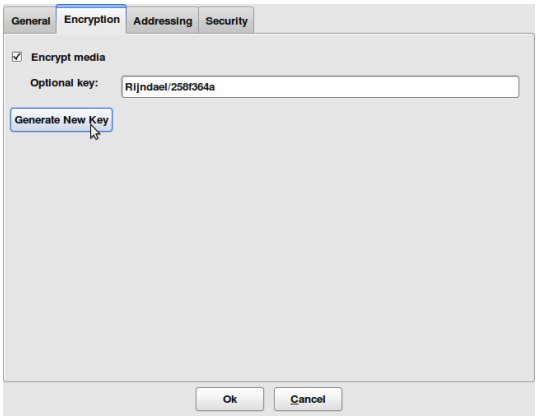


Como se comentó en la sección *Configuración General de un Servidor de Salas*, Access Grid, por cuestiones de seguridad, nos permite encriptar todo el flujo de información que se transporte en una sesión.

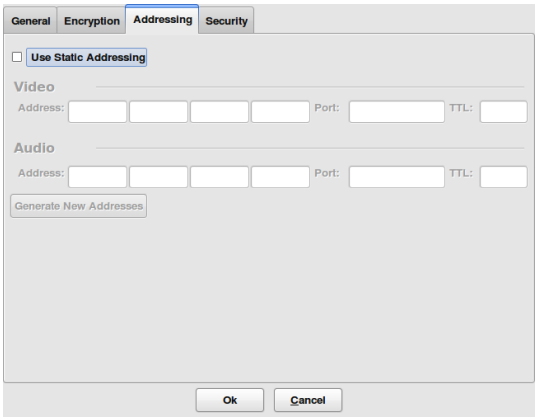
Esta pestaña es, prácticamente, la misma que en dicha sección. La diferencia está en que la sección anterior, el cambio afecta a todas las Salas que existan en nuestro Servidor de Salas. En cambio, los cambios que realicemos en esta pestaña afectarán únicamente a la Sala seleccionada.

En esta pestaña nos aparecen las siguientes opciones:

- **Encrypt media:** Si activamos esta casilla estamos indicando a Access Grid que encripte todo flujo multimedia que exista en la Sala seleccionada, aumentando así la seguridad de la Sala. Una vez activada la casilla se nos habilita una nueva opción:
  - **Optional key:** Aquí introducimos la clave que se utilizará para encriptar el contenido multimedia. Si no sabemos cómo hacerlo o, para mayor facilidad, podemos pulsar el botón **Generate New Key**, que nos generará automáticamente la clave.



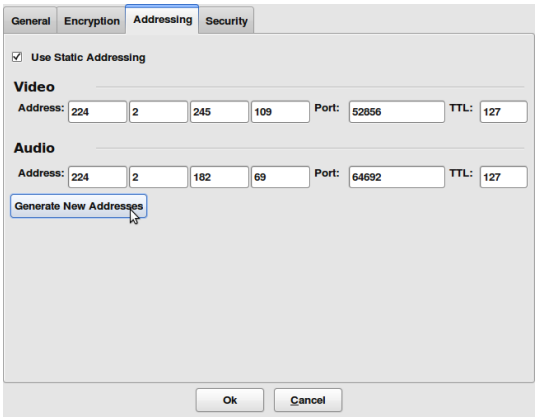
Addressing



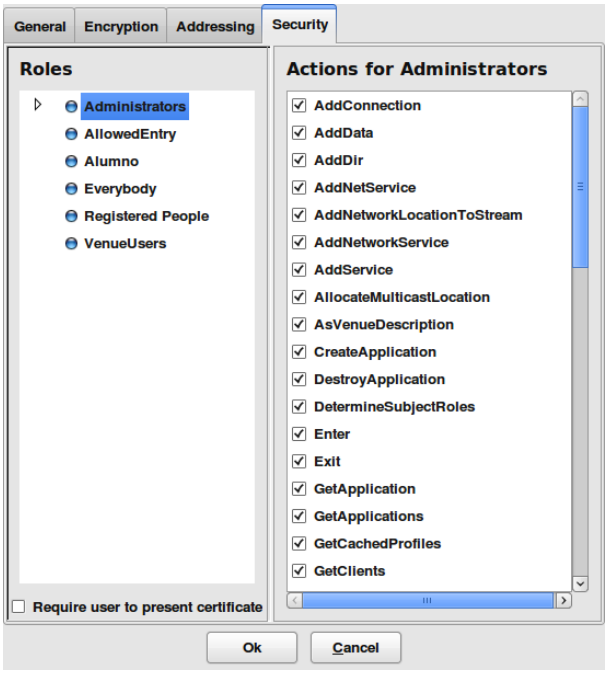
En esta pestaña se recogen el conjunto de opciones relacionadas con el direccionamiento de los flujos de audio y vídeo. En algunas ocasiones es necesario establecer un direccionamiento estático, por ejemplo, cuando una empresa tiene configurada la red de forma que cualquier dirección y puerto están cerrados y únicamente tienes disponible una determinada dirección IP y puerto.

Las opciones que nos aparecen en esta pestaña son:

- **Use Static Addressing:** Si activamos esta casilla estamos indicando a Access Grid que queremos utilizar direccionamiento estático para los flujos de audio y vídeo de la Sala seleccionada. Una vez activada se nos habilitan las siguientes opciones:
  - **Video Address|Port|TSL:** Introducimos la dirección IP, el puerto y el tiempo de vida para el flujo de vídeo
  - **Audio Address|Port|TSL:** Introducimos la dirección IP, el puerto y el tiempo de vida para el flujo de audio
  - **Generate New Addresses:** Si lo único que queremos es establecer un direccionamiento estático a los flujos de audio y vídeo, sin importar las direcciones IPs y puertos, pulsando este botón se nos generará automáticamente dichas IPs, puertos y flujos de vida para el audio y para el vídeo.



Security



Esta pestaña es muy similar a la pestaña Security dentro de la sección *Configuración General de un Servidor de Salas*, con la diferencia de que, en la sección anterior, se limitaba el control de acceso para acceder a un Servidor de Salas para su configuración, a través del propio Gestor de Salas (*Venue Management*) y, en esta pestaña, limitamos el control de acceso a una Sala determinada.

Como en la sección anterior, en esta pestaña podemos gestionar el acceso a una Sala determinada, permitiendo/denegando el acceso a personas para acceder a una Sala, o limitando/ampliando sus permisos, como poder añadir archivos a la Sala, poder iniciar una aplicación compartida, poder visualizar vídeo o audio, etc. Es decir, por defecto, cuando creamos una Sala, todo el mundo puede acceder a ella y tiene todos los permisos, sin embargo, en la mayoría de los casos, esto no es deseable (por ejemplo, no es deseable que un alumno de Enfermería acceda a una Sala de Mecánica), por lo que tendremos que utilizar esta herramienta para controlar el acceso y los permisos de cada participante, a través de los **certificados digitales**.

Para ello existen un conjunto de **Roles, Participantes y Acciones** (permisos). Podemos crear/eliminar Roles y crear/eliminar Participantes. Por cada Rol podemos activar/desactivar Acciones. Por último, podemos asignar a los participantes uno o varios Roles.

Por ejemplo, podemos crear el Rol **Profesor**, que tiene menos permisos que un **Administrador**, y, además, podemos crear el Rol **Alumno**, que tiene aún menos permisos que un Profesor.

A.4.5. Control de Acceso

Como se ha explicado en los apartados de *Security* tanto en la sección de *Configuración General de un Servidor de Salas* como en la sección de *Edición de Salas*, Access Grid incorpora un sistema de control de acceso, tanto para el Servidor de Salas, como para una Sala concreta. En él, podremos crear

roles y personas (participantes) y asignarles o quitarles permisos. A continuación se explicará los pasos a seguir para ello:

Certificados Digitales

Access Grid controla el acceso y los permisos de los usuarios a través de los **certificados digitales**[3]. De forma resumida, un certificado digital es un documento digital donde una **entidad certificadora** garantiza la identificación del usuario a través de dicho documento. Existen varios tipos de certificados digitales, sin embargo, los que usaremos en Access Grid son los certificados **x509** [4].

La forma de hacer referencia a un certificado digital es a través de su ***Distinguished Name (DN)***. La autoridad certificadora (AC) emite el certificado una clave pública a un DN. El DN es la identificación única de un certificado, se compone de varios atributos. La forma de un DN es una secuencia de pares *atributos=valor* separados por coma o por un espacio y el orden en el que aparecen los pares es importante. Estos atributos reciben el nombre de ***Relative Distinguished Names (RDN)***. Estos atributos son:

- **DC:** domainComponent. Componente del dominio
- **CN:** commonName. Nombre común
- **OU:** organizationalUnitName. Nombre de unidad de la organización
- **O:** organizationName. Nombre de la organización
- **STREET:** streetAddress. Domicilio
- **L:** localityName. Localidad
- **ST:** stateOrProvinceName. Provincia
- **C:** countryName. País
- **UID:** userId. Identificación del usuario

Por tanto, combinando estos atributos podemos generar nuestro DN. Por ejemplo, un DN podría ser:

```
1 /O=Access Grid/OU=agdev-ca.mcs.anl.gov/OU=agserver.uca.es/CN=Jesus Cea
     Oliva
```

Así pues, Access Grid, utiliza los DN para identificar a los participantes que se conectan al Servidor de Salas y, a partir de él, conceder o denegar permisos.

Pasos a seguir para controlar el acceso a un participante al Servidor de Salas

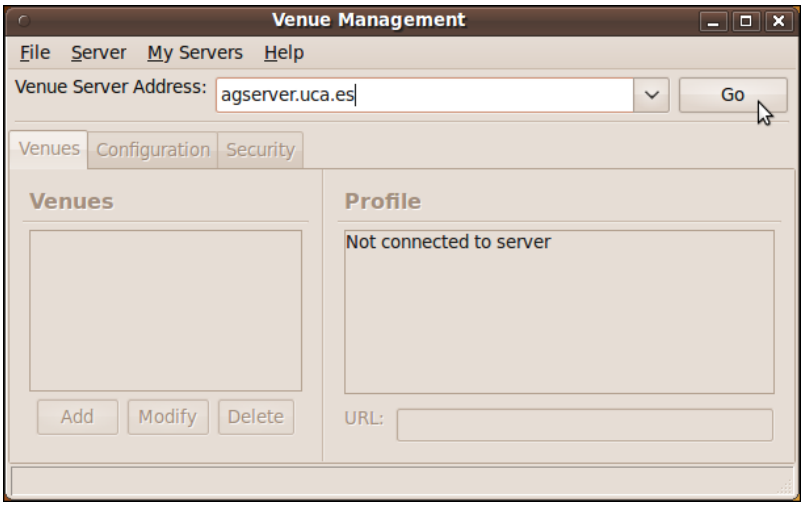
En este apartado se detallarán los pasos a seguir para controlar los permisos a un participante concreto. Utilizaremos, como ejemplo, el DN mencionado anteriormente, es decir:

```
1      /O=Access Grid/OU=agdev-ca.mcs.anl.gov/OU=agserver.uca.es/CN=Jesus Cea
          Oliva
```

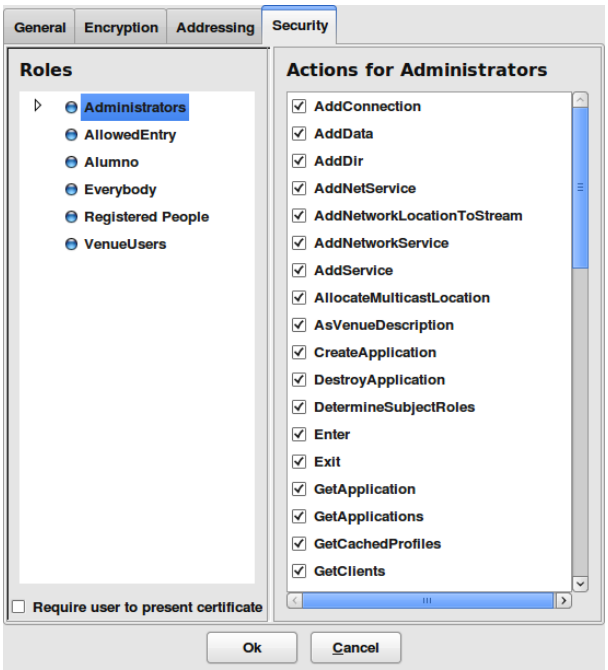
[3] http://es.wikipedia.org/wiki/Certificado_digital
[4] http://es.wikipedia.org/wiki/X.509

Supongamos que queremos darle permisos como Administrador en el Servidor de Salas. Los pasos a seguir son:

- Desde el PC del Servidor de Salas ejecutamos el Gestor de Salas (*VenueManager*). Introducimos la dirección del Servidor de Salas y pulsamos el botón **Go**:



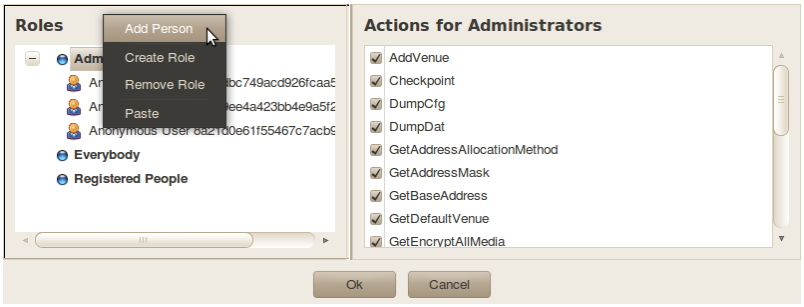
- Como lo que queremos es asignarle permisos al usuario para poder configurar el Servidor de Salas, nos dirigimos directamente a la pestaña **Security** y pulsamos el botón **Manage Security**. Si quisiéramos asignarles permisos a una Sala concreta, elegimos la Sala entre las Salas que aparece en el listado, pulsamos el botón **Modify**, y nos dirigimos a la pestaña **Security**. Nos aparecerá una ventana como ésta:



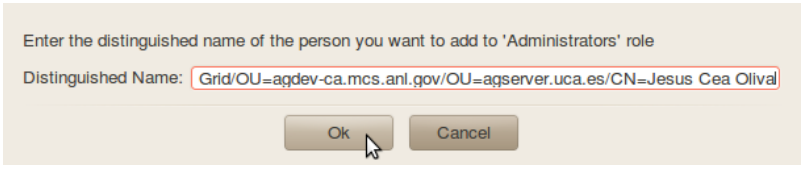
- Recordemos que en el lado izquierdo aparece un listado de **participantes** agrupados por **roles** y, a la derecha, aparece un listado de permisos que aparecerán activados o desactivados según el rol

elegido.

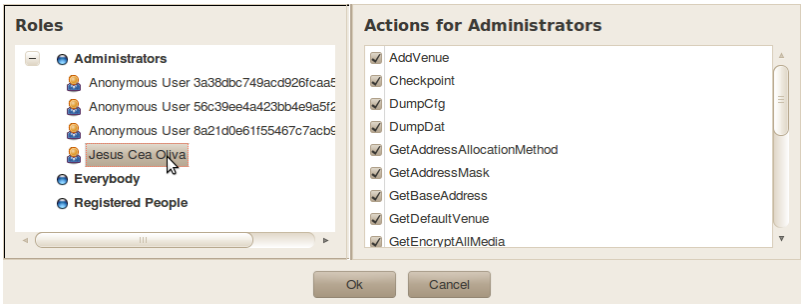
- Si pulsamos el segundo botón en el listado izquierdo, nos aparecen las siguientes opciones:
  - **Add Person:** Para añadir un nuevo participante.
  - **Remove Person** (previamente debemos haber pulsado el segundo botón en una persona seleccionada): Para eliminar al participante seleccionado.
  - **Create Role:** Para crear un nuevo Rol.
  - **Remove Role** (previamente debemos haber pulsado el segundo botón en un rol): Para eliminar el rol seleccionado.
- Así pues, para añadir al nuevo usuario, pulsamos el segundo botón en el rol al que queremos asignarle, en este caso, al rol **Adminstrators**, y seleccionamos **Add Person**. Aún así, si, por algún motivo, nos equivocamos de Rol, una vez creado el participante, podremos arrastrarlo al rol deseado.



- Nos aparecerá una ventana solicitándonos el ***Distinguished Name*** del participante. Introducimos el DN del ejemplo



- Una vez pulsemos el botón **Ok**, nos aparecerá el usuario dentro del grupo de Administradores:



- Ahora, el usuario, desde cualquier PC donde tenga instalado el certificado correspondiente a ese DN, podrá acceder a la configuración del Servidor de Salas como administrador.

- Como ya se ha comentado, si, por algún motivo, quisiera cambiar el rol del usuario creado, simplemente lo podemos arrastrar al rol que queremos asignarle.

- Aunque se ha comentado como ejemplo crear un usuario como Administrador, puede repetir estos pasos para otro rol cualquiera y otro usuario cualquiera. Por otro lado, **un usuario puede tener varios roles**, por lo que podremos copiar los participantes y pegarlos en cualquier rol.

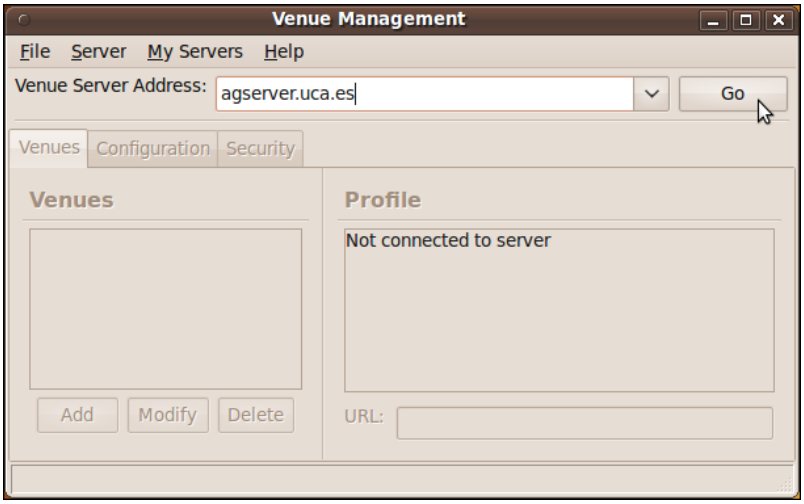
Pasos a seguir para controlar el acceso a un participante a una Sala

Para controlar el acceso a una Sala a un participante dado, hacemos lo siguiente, teniendo en cuenta que utilizaremos, como ejemplo, el DN mencionado anteriormente, es decir:

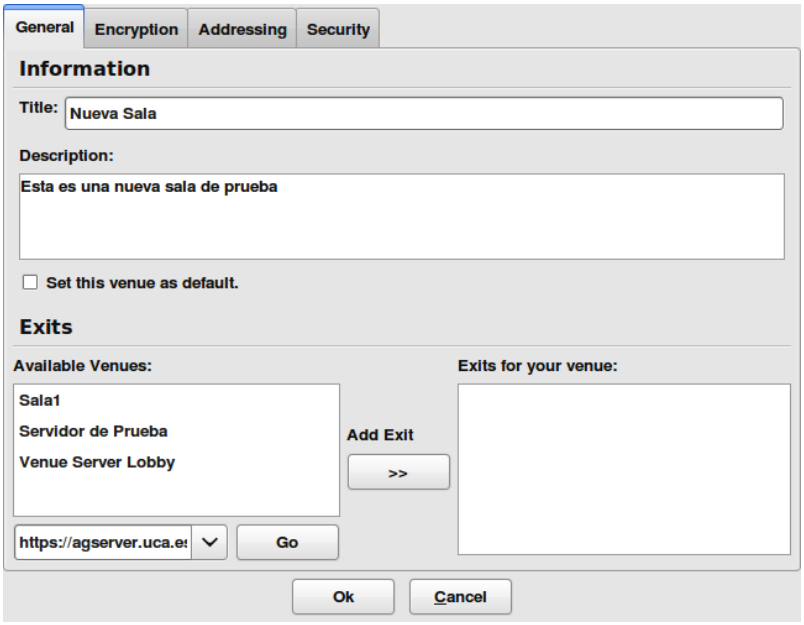
```
1       /O=Access Grid/OU=agdev-ca.mcs.anl.gov/OU=agserver.uca.es/CN=Jesus Cea
           Oliva
```

Supongamos que queremos darle permisos como Administrador en la Sala **Venue Server Lobby**. Los pasos a seguir son:

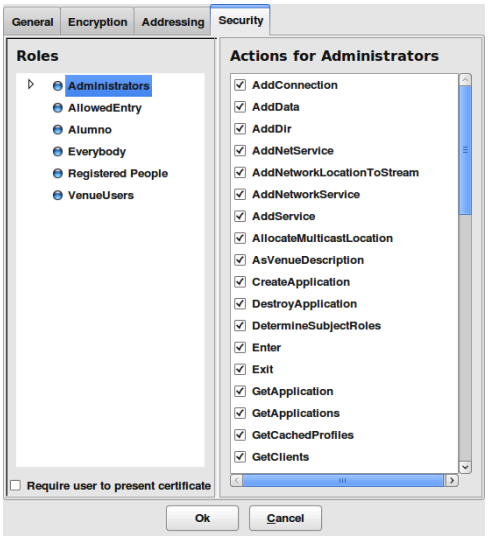
- Desde el PC del Servidor de Salas ejecutamos el Gestor de Salas (*VenueManager*). Introducimos la dirección del Servidor de Salas y pulsamos el botón **Go**:



- Como lo que queremos es asignarle permisos al usuario para una Sala concreta, en este caso, la Sala **Venue Server Lobby**, seleccionamos dicha sala y pulsamos el botón **Modify**. Nos aparecerá la siguiente ventana (puede tardar en aparecer):



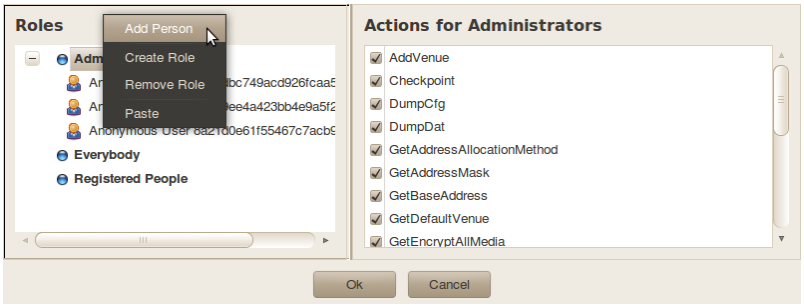
- Nos dirigimos a la pestaña **Security** y nos aparecerá la siguiente ventana:



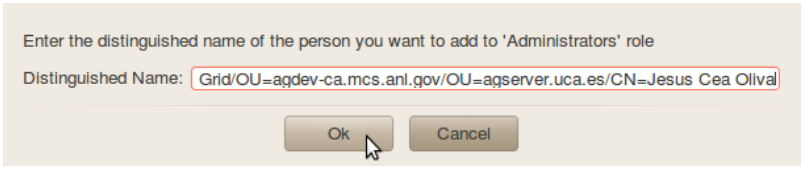
- Recordemos que en el lado izquierdo aparece un listado de **participantes** agrupados por **roles**, y, a la derecha, aparece un listado de permisos que aparecerán activados o desactivados según el rol elegido.
- Nos aseguramos que está marcada la opción **Require user to present certificate**.
- Si pulsamos el segundo botón en el listado izquierdo, nos aparecen las siguientes opciones:
  - **Add Person:** Para añadir un nuevo participante.
  - **Remove Person** (previamente debemos haber pulsado el segundo botón en una persona seleccionada): Para eliminar al participante seleccionado.
  - **Create Role:** Para crear un nuevo Rol.

- **Remove Role** (previamente debemos haber pulsado el segundo botón en un rol): Para eliminar el rol seleccionado.

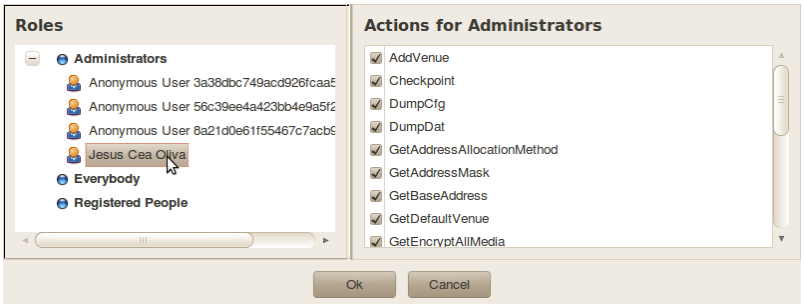
- Así pues, para añadir al nuevo usuario, pulsamos el segundo botón en el rol al que queremos asignarle, en este caso, al rol **Adminstrators**, y seleccionamos **Add Person**. Aún así, si, por algún motivo, nos equivocamos de Rol, una vez creado el participante, podremos arrastrarlo al rol deseado.



- Nos aparecerá una ventana solicitándonos el ***Distinguished Name*** del participante. Introducimos el DN del ejemplo:



- Una vez pulsemos el botón **Ok**, nos aparecerá el usuario dentro del grupo de Administradores:



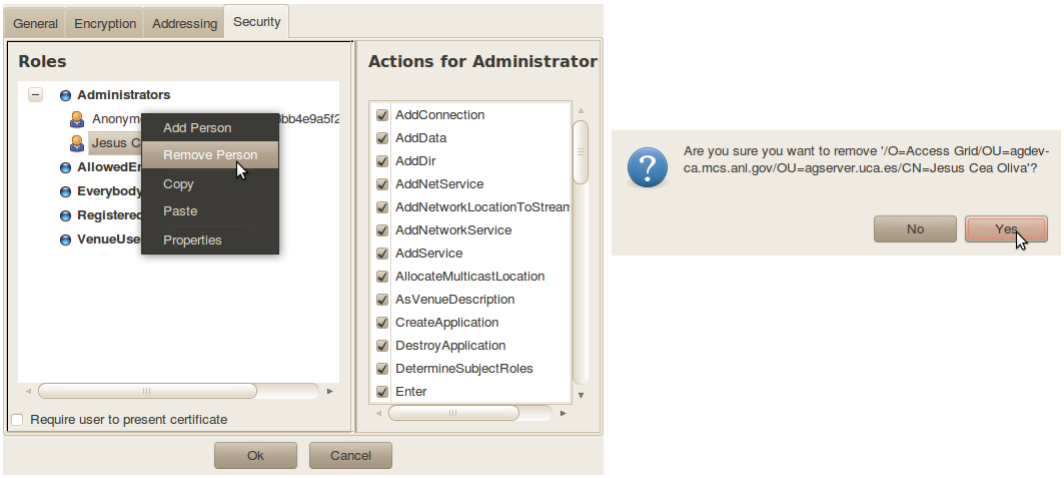
- Ahora, el usuario podrá ser administrador de la Sala **Venue Server Lobby**.

- Como ya se ha comentado, si, por algún motivo, quisiera cambiar el rol del usuario creado, simplemente lo podemos arrastrar al rol que queremos asignarle.

- Aunque se ha comentado como ejemplo crear un usuario como Administrador, puede repetir estos pasos para otro rol cualquiera y otro usuario cualquiera. Por otro lado, **un usuario puede tener varios roles**, por lo que podremos copiar los participantes y pegarlos en cualquier rol.

Borrar un participante

Para borrar a un participante, lo seleccionamos de la lista de la izquierda, pulsamos el segundo botón y seleccionamos **Remove Person**. Nos preguntará si estamos seguros y, si lo estamos, pulsamos el botón **Yes**.



Crear un Rol

Para crear un nuevo rol, pulsamos el segundo botón y seleccionamos **Create Role**, nos solicitará un nombre, así que lo introducimos y pulsamos el botón Ok. Luego activamos o desactivamos aquellos permisos que queremos asignarles a dicho rol.

Borrar un Rol

Para borrar un rol, seleccionamos dicho rol y pulsamos el segundo botón, seleccionamos **Remove Role**. Nos preguntará si estamos seguros de que queremos eliminarlo y, si lo estamos, pulsamos el botón **Yes**.

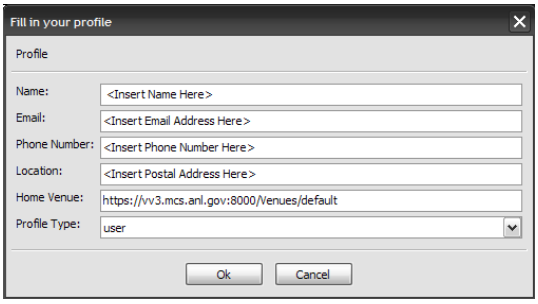
A.5. El Cliente de Sala

El Cliente es el software encargado de conectarse y participar en una Sala Virtual de Access Grid. En él, se visualizan los participantes conectados actualmente en una Sala concreta, otras conexiones para acceder a otras Salas, los contenidos que dispone la Sala, además de una interfaz para su configuración.

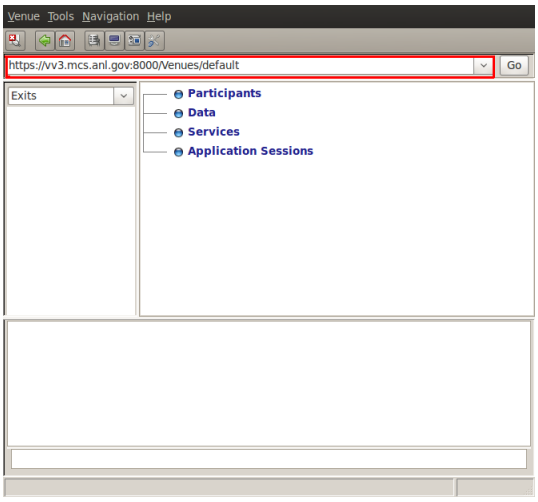
El procedimiento a seguir es el mismo tanto en Windows como en Linux.

- Ejecutamos el Cliente Access Grid.
  - **Linux:** Para ello vamos a ”Aplicaciones− >Access Grid 3− >Venue Client” (también podemos ejecutar desde un terminal la orden *VenueClient3.py*)
  - **Windows:** Podemos o bien ejecutar la aplicación *Venue Client* desde el acceso directo desde el escritorio o, en caso de que no exista o lo haya borrado, desde el ”Menú de Inicio− >Todos los programas− >Access Grid 3− >Venue Client”

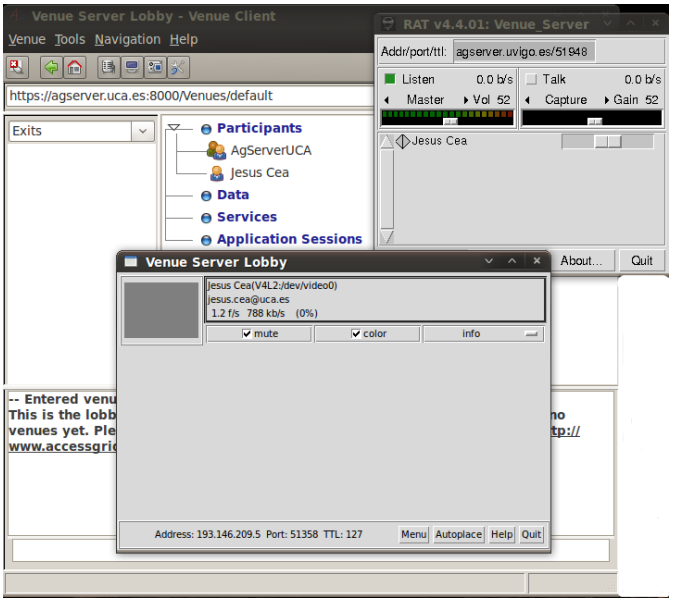
- La primera vez que se ejecute el programa le aparecerá una pantalla solicitando datos de interés, rellénela con sus datos. En el apartado *Profile Type* seleccione **user**.



- Una vez ejecutada la aplicación, escribimos la URL del Servidor de Salas al que queremos acceder y pulsamos el botón **Go** para acceder.



- A continuación, cuando se acceda al Servidor de Salas, aparecerá su nombre de usuario como participante de la sala (sección *Participants*). Además, dependiendo de la configuración (si se ha creado una o por defecto), se cargarán los correspondientes servicios de audio y vídeo.



- En el caso de que no aparezca ningún servicio, asegúrese de que están activadas las opciones de la sección *Tools*:
  - Enable Audio
  - Enable Video Display
  - Enable Video Capture

A.5.1. Servicios

Access Grid, entre otras funciones, gestiona la recepción y emisión tanto de sonido como de vídeo a través de **servicios**. Los servicios son, básicamente, contenedores de software que pueden ejecutarse en un Cliente de Access Grid (VenueClient) cuando dicho Cliente accede a una Sala. Los servicios pueden ser de diversos tipos, incluso, si se desea, se pueden diseñar servicios personalizados, empaquetarlos, distribuirlos y cargarlos en el Cliente para añadir funcionalidades.

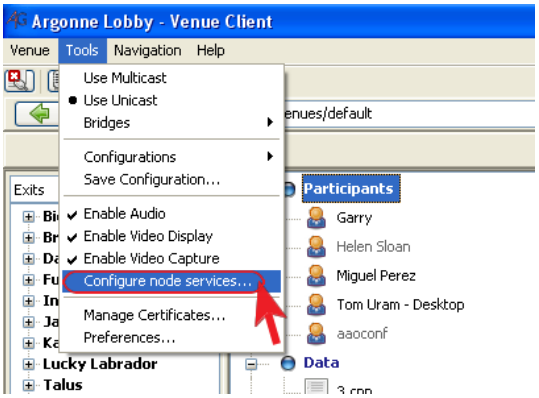
Los servicios más comunes que hay son:

- **AudioService:** Para emitir y recibir el sonido a través de la aplicación *RAT*.
- **VideoService:** Para emitir y recibir el vídeo a través de la aplicación *Vic*.
- **VideoConsumerService:** Para únicamente recibir el vídeo (sin emitir). Opción que deben utilizar aquellos PCs que no dispongan de WebCam.
- **VideoProducerService:** Para únicamente emitir el vídeo (sin recibir).
- **Otros Servicios:** Dependiendo de los paquetes que hayamos instalado, pueden aparecer nuevos servicios, como por ejemplo VideoServiceH264 para la retransmisión y emisión de vídeo utilizando el códec H.264, o HDVideoService para la retransmisión y emisión de vídeo en HD.

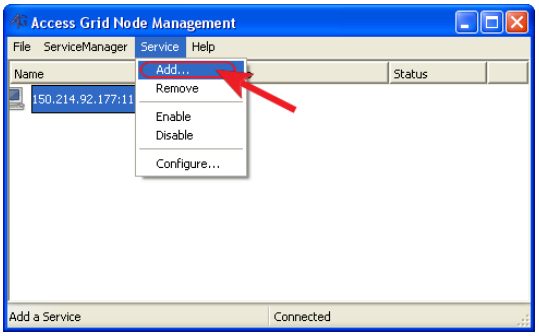
La configuración es prácticamente la misma bajo las distintas plataformas (Windows y Linux). A continuación se detalla cómo hacerlo.

Añadir Servicios en Windows XP

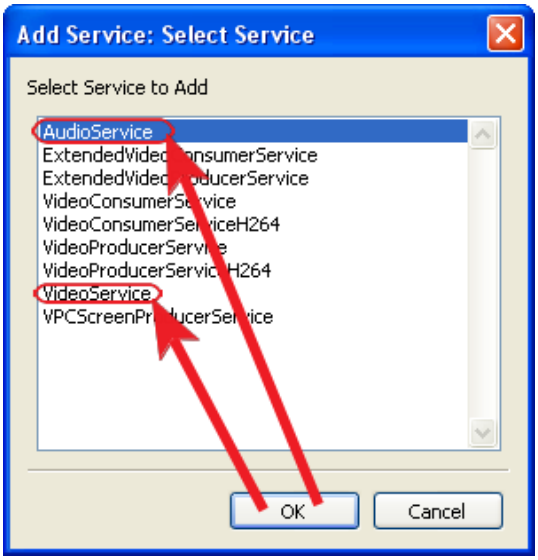
- Una vez arrancado el cliente de Access Grid nos dirigimos al menú ”Tools” y hacemos click en la opción **Configure node service...**:



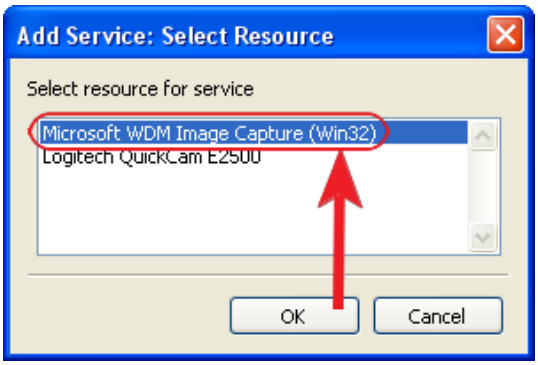
- Se nos abrirá una ventana llamada ”Access Grid Node Management”. Pinchamos en el menú ”Service” y luego en ”Add...”:



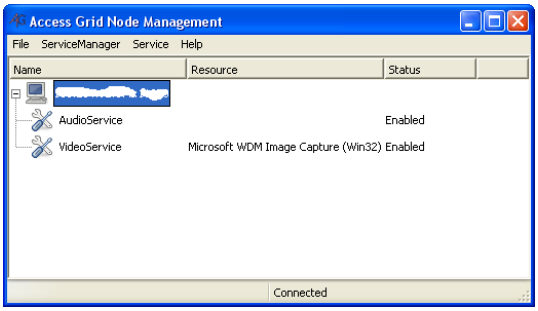
- Seleccionamos **AudioService** y pinchamos en OK. Luego seguimos el mismo paso anterior para agregar **VideoService**. *NOTA: VPCScreenProducerServer sirve para retransmitir nuestro escritorio como si fuera una señal de vídeo.*



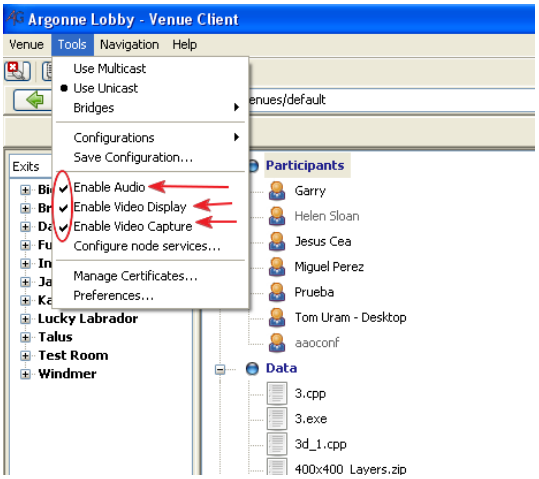
- Al seleccionar *VideoService* y pulsar OK, nos aparecerá con casi toda seguridad una nueva ventana para seleccionar el origen de vídeo. En la página oficial de Access Grid se recomienda usar **Microsoft WDM Image Capture (WIN32)**, así que lo seleccionamos (si no nos funciona esta opción o no aparece, seleccionamos nuestro origen de vídeo):



- Una vez hecho esto la ventana *Access Grid Node Managemente* debería quedarnos tal que así:

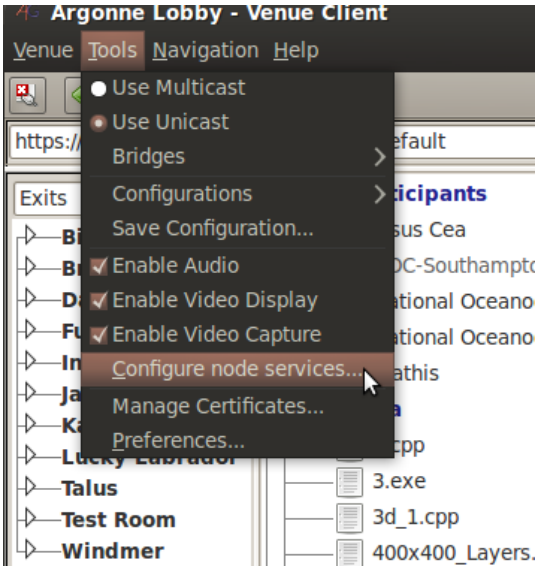


- Ahora el audio y el vídeo deberían estar emitiendo y recibiendo correctamente. Si no es así, vamos al menú ”Tools” y nos aseguramos de que las opciones *Enable Audio*, *Enable Video Display* y *Enable Video Capture* están marcadas como en la siguiente imagen:

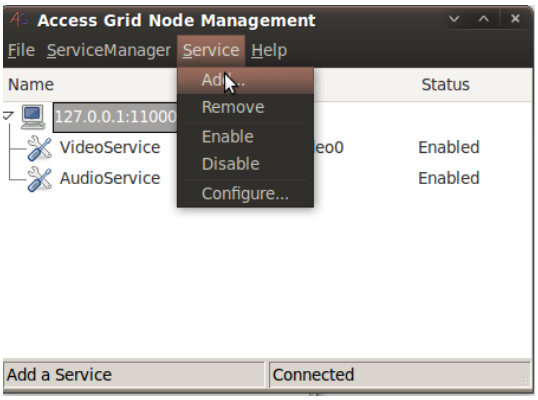


Añadir Servicios en Ubuntu

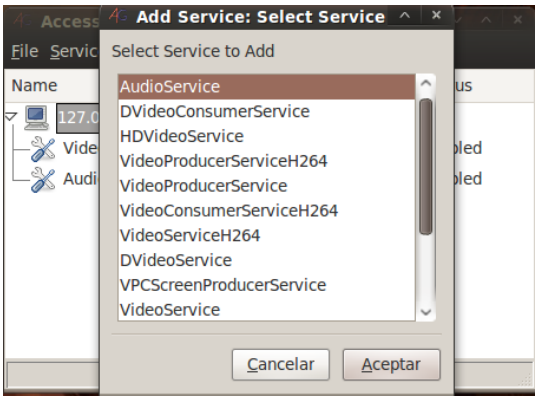
- Una vez ejecutado el cliente de Access Grid, nos dirigimos a *Tools* $->$ *Configure node services...*:



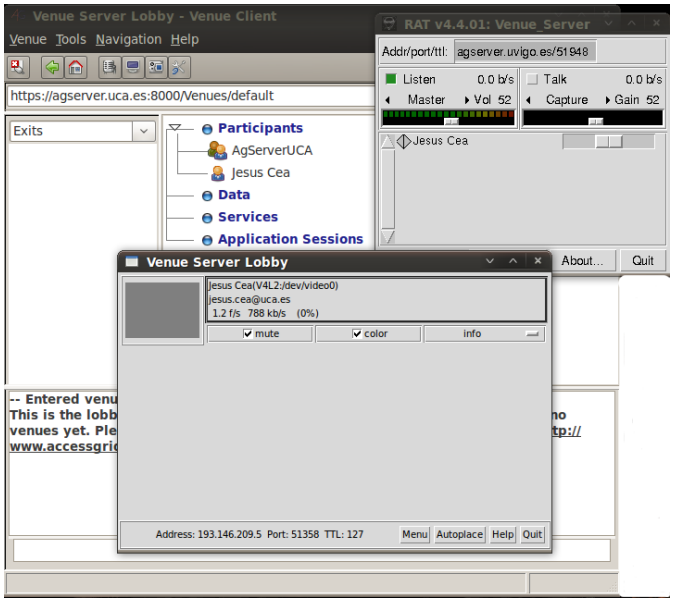
- En la siguiente ventana que nos aparecerá seleccionamos "Service $->$ Add":



- Nos aparecerá un listado de Servicios. Seleccionamos aquél o aquéllos que necesitemos.

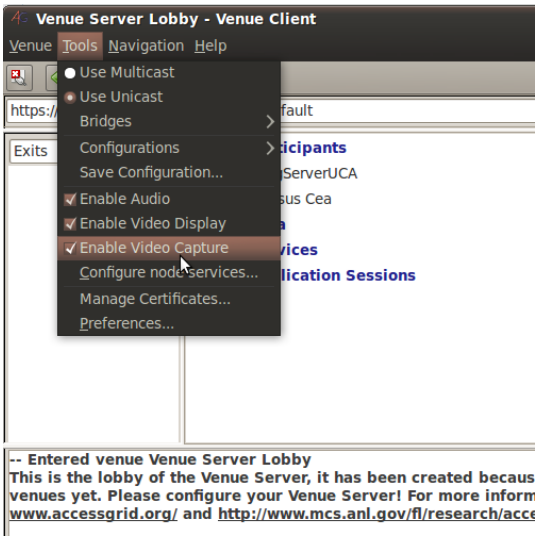


- Por último, aparecerán las aplicaciones correspondientes a los servicios seleccionados. Para el audio, se ejecutará la aplicación *RAT* y para el vídeo la aplicación *Vic*.



- En cualquier caso, hay que asegurarse de que están activadas las opciones de la sección ”Tools”:

  - Enable Audio
  - Enable Video Display
  - Enable Video Capture



Añadir servicios situados en otra máquina

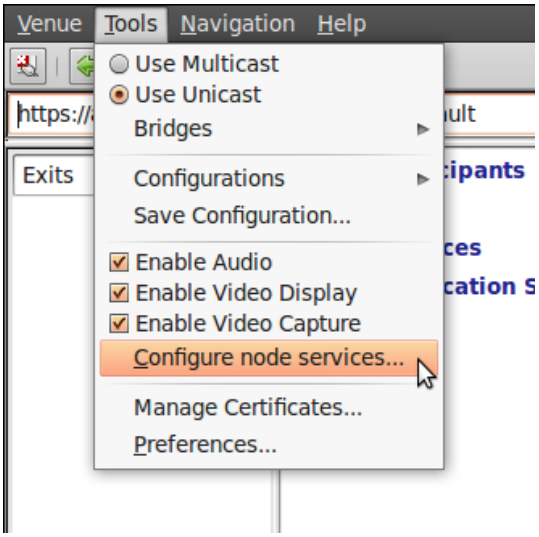
Access Grid permite añadir Servicios que estén alojados en otra máquina y configurarlos remotamente. Este tipo de configuración en Access Grid se conoce como **Configuración Multi-Nodo**. La herramienta que permite esta funcionalidad se llama **ServiceManager**, la cual, ejecuta un servicio en el que, por un lado, ”captura” todas las funcionalidades que la máquina, donde se esté ejecutando el ServiceManager, dispone (cámaras de vídeo, compartir el escritorio, audio, aplicaciones compartidas, etc.) y, por otro lado, abre una vía de comunicación a la espera de que un Cliente de Access Grid solicite los servicios capturados.

Una de las principales ventajas que tiene este tipo de configuración es que, al tratarse de una configuración Multi-Nodo, la carga de trabajo se reparte entre las diferentes máquinas que forman parte de dicho Multi-Nodo. Así, una máquina se puede dedicar a capturar vídeo y enviarlo a la Sala de Access Grid, otra recibe toda fuente de vídeo y las muestras al usuario (visualizador) e incluso puede haber una tercera máquina dedicada exclusivamente a la gestión del audio.

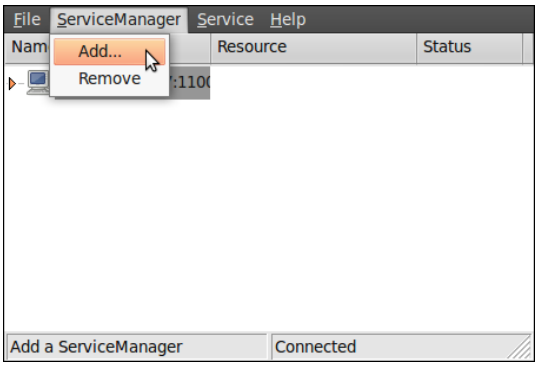
En nuestro caso, disponemos de dos máquinas dedicadas a Access Grid. Por un lado tenemos una máquina dedicada a la captura de vídeo y su emisión, además de gestionar el audio, y, la otra máquina, tiene la función de visualizador, recibiendo todas las señales de vídeo que envían en la Sala para mostrarlas luego al usuario.

La forma de añadir un ServiceManager a nuestro Cliente de Sala es la siguiente:

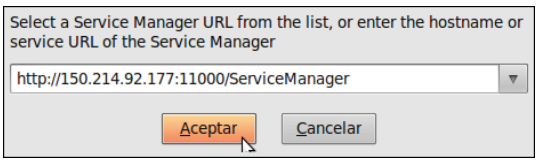
- Desde la ventana del *Venue Client* nos dirigimos al menú ”Tools $->$ Configure node services...”:



- En la nueva ventana vamos a ”ServiceManager − > Add...”:



- Introducimos la IP de la máquina siguiendo el formato *IP* + *:* + + *puerto (normalmente el 11000)* + *ServiceManager*, y pulsamos ”Aceptar”. Como en la imagen siguiente:



- Después es posible que tengamos que añadir los servicios manualmente como si fueran servicios locales.

A.6. Configuración de Vídeo (*Vic*)

En una videoconferencia, el vídeo es una de las claves de éxito de ésta y es muy importante el que exista una buena calidad puesto que, al tratarse de una videoconferencia, los participantes pueden verse entre ellos consiguiéndose, así, simular como si una reunión presencial real se tratara.

En este apartado se intentará dar unas simples nociones relacionadas con el vídeo, con el objetivo de que pueda conseguir la mejor relación calidad de vídeo/rendimiento posible en una reunión.

En general, existen dos tipos de servicios de vídeo, dependiendo de la funcionalidad deseada:

- **Servicio de Visualizador** (*VideoConsumer* y variantes): Servicio cuya función es recibir todas las fuentes de vídeo que llegan de la Sala.
- **Servicio de Captura y Emisión** (*VideoProducer* y variantes): Servicio cuya función es capturar la/s fuente/s de vídeo y transmitirlas a la Sala.

Se explicará brevemente el uso y configuración de la aplicación Vic (*Video Conference Tool*), la aplicación dedicada a la emisión y recepción de vídeo en cada uno de estos ámbitos. Además, se hablará de un problema de rendimiento con dicha aplicación en casos especiales, se explicará el motivo y las posibles soluciones.

A.6.1. Configuración de Vic como Visualizador (*VideoConsumer* y variantes)

La aplicación Vic (*Video Conference Tool*) viene integrada en el sistema Access Grid. Cuando entramos en una sesión de Access Grid, si tenemos añadido, o queremos añadir, un servicio de vídeo, bien sea productor o visualizador, se ejecutará automáticamente para gestionar el vídeo, tanto la emisión, como la recepción, dependiendo del servicio añadido. En esta sección se explicará cómo configurar *Vic* como **Visualizador** (VideoConsumer y variantes).

Cuando entramos a una Sala de Access Grid, si tenemos añadido el servicio de Visualización de vídeo, nos aparecerá una instancia de la aplicación Vic donde se listan todas las fuentes de vídeo que se están retransmitiendo actualmente en dicha sala:

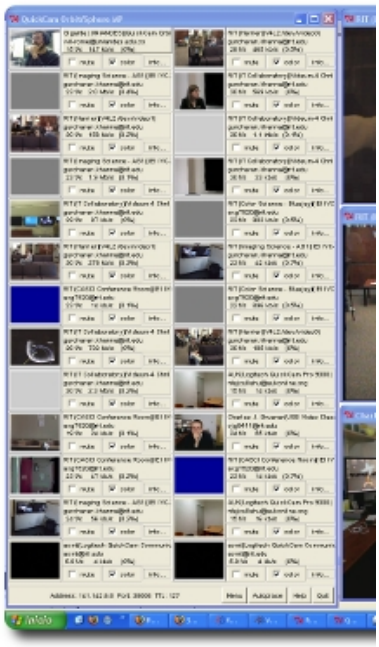
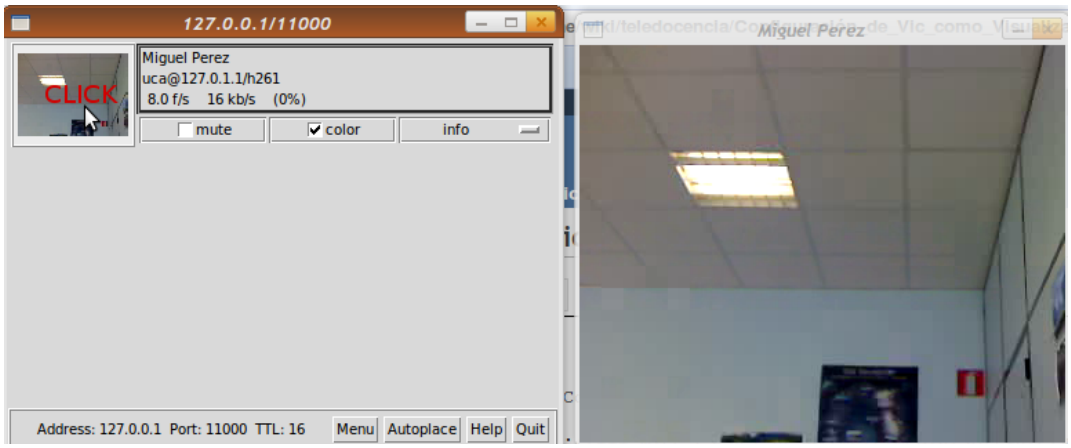


Figura A.18: Ventana de la aplicación Vic

Este listado muestra una vista previa de cada fuente de vídeo, por lo que la velocidad de imágenes por segundo y su calidad **no es la real**. Para ver la fuente de vídeo tal y como es, tenemos que hacer click en su correspondiente vista previa, nos aparecerá una ventana con la fuente de vídeo retransmitiéndose:



Dicha ventana podemos cambiar su tamaño pulsando una tecla, dependiendo del tamaño que queremos establecer:

- **L** (*Large*): Si pulsamos la tecla L estableceremos el tamaño de ventana a Grande.
- **M** (*Medium*): Si pulsamos la tecla M estableceremos el tamaño de ventana a Normal/Medio.
- **S** (*Small*): Si pulsamos la tecla S estableceremos el tamaño de ventana a Pequeño.

Podemos configurar Vic de tres formas distintas:

- A través del Gestor de Servicios (*Tools* $->$*Configure node services*), que será la recomendada, ya que, los cambios que hagamos desde aquí, podrán ser guardados en un fichero de configuración para no tener que repetir el procedimiento. Además, desde aquí podemos configurar los servicios que estén alojados en otra máquina (a través de ServiceManager, ver *Añadir Servicios situados en otra máquina*), y, como antes, también podremos guardar la configuración en un fichero de configuración.
- Desde el propio Vic: Se configura directamente desde la propia aplicación. Recomendado para configuraciones concretas, ya que los cambios que se haga no podrán ser guardados en un fichero de configuración. Sin embargo, al tratarse de la propia aplicación, permite una configuración más avanzada que la que ofrece el Gestor de Servicios.
- Crear/Editar un fichero de configuración: Access Grid permite guardar ficheros de configuración, donde se almacenan las configuraciones de los servicios que usemos (audio, vídeo, etc.) en nuestra sesión de Access Grid. Si uno de los servicios que usamos es un servicio de vídeo (ya sea visualizador o productor), podemos editar nuestro fichero para configurar aquellas opciones que ofrece Vic pero no están disponibles desde el Gestor de Servicios. En capítulos posteriores se explicará con detalle cómo editar o crear un fichero de configuración personalizado.

Como en esta sección se explicará Vic como Visualizador, hay que asegurarse de que hemos añadido el servicio VideoConsumerService o alguna de sus variantes (VideoService, VideoConsumerServiceH264, etc.). Para añadir el servicio vea el capítulo anterior, las secciones:

- **Añadir Servicios en Windows XP:** Si su sistema es un sistema Windows
- **Añadir Servicios en Ubuntu:** Si su sistema es Ubuntu o cualquier otro sistema Unix

Si se va a configurar a través del Gestor de Servicios del Nodo de Access Grid, deberá saber que en la versión de Windows se incluye una opción extra. Se explicará a continuación.

Configuración a través del Gestor de Servicios (Versión Unix)

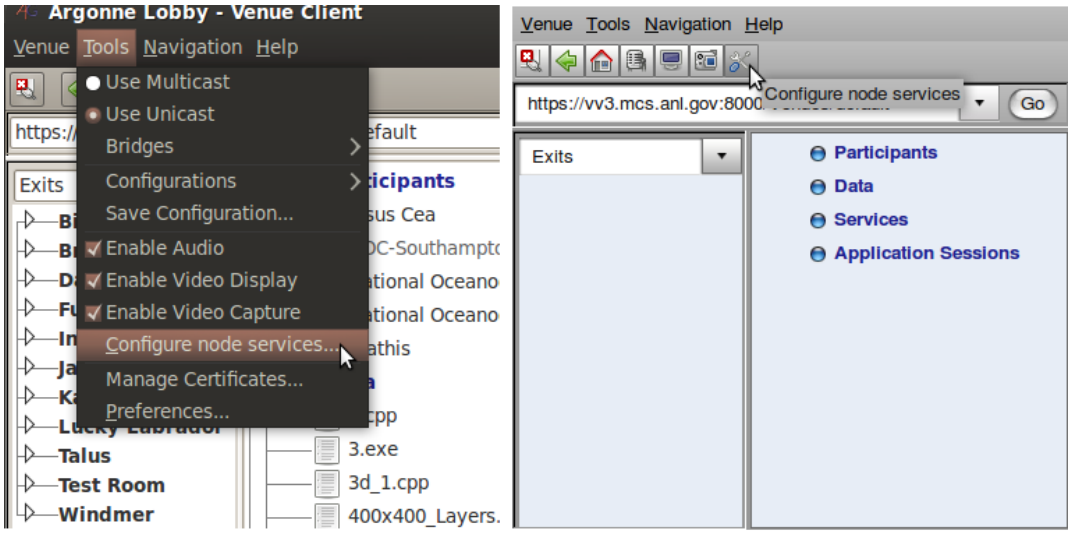
Uno de los métodos para poder configurar Vic es a través del **Gestor de Servicios del Nodo de Access Grid**. Vic es una aplicación independiente que ha sido integrada en el sistema Access Grid, así que, desde este método, nos aparecerá una interfaz sencilla con las opciones que se han considerado relevantes para el buen funcionamiento del Vídeo en Access Grid.

Sin embargo, utilizar este método de configuración tiene el inconveniente de que los cambios que hagamos en la configuración no surtirán efecto a no ser que, o bien deshabilitemos y habilitemos el servicio de Video como **Visualizador** (*VideoConsumer* y variantes), o bien, volviendo a entrar en la Sala.

Por similitud de las diferentes variantes existentes del Servicio de Visualización de Vídeo (*VideoConsumerService*, *VideoConsumerServiceH264*, etc), nos basaremos en el servicio estándar que ofrece Access Grid, éste es el servicio *VideoConsumerService*.

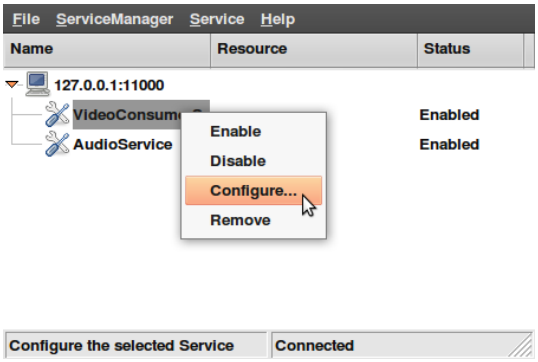
Para configurar Vic como visualizador a través del Gestor de Servicios, hacemos lo siguiente:

- En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:

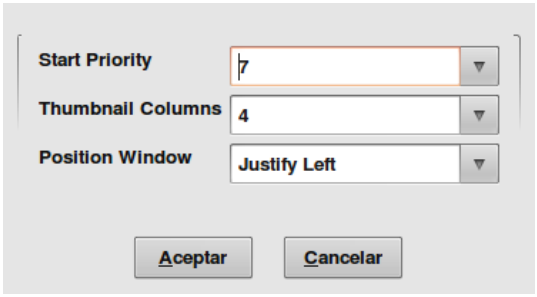


- Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Hacemos doble click en **VideoConsumerService**, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:

  Nota: Hay que recordar que hay que tener añadido el servicio VideoConsumerService, en caso de que no aparezca en el listado, hay que añadirlo.



- Nos aparecerá la pantalla de configuración del visualizador de vídeo:



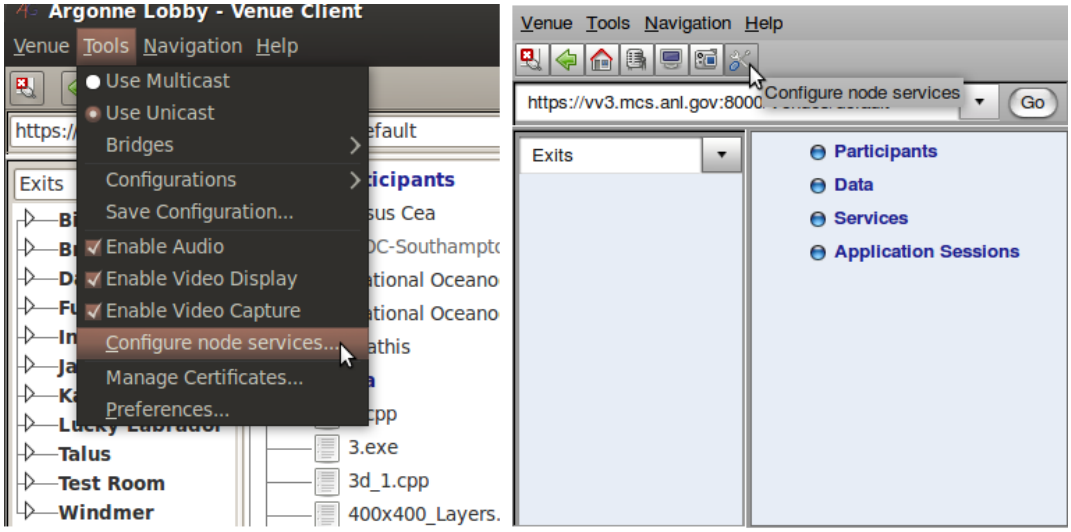
- Las opciones que podemos encontrar son:

  - **Start Priority** [**Valores:** *1 al 10*, **Por defecto:** *7*]: Esta opción nos permite variar la prioridad de este proceso, a mayor prioridad, más uso de CPU se le asigna.
  - **Thumbnail Columns** [**Valores:** *1 al 10*, **Por defecto:** *4*]: Esta opción modifica la distribución de las ventanas de vídeo que Vic va recibiendo de la Sala. El valor indica el número de columnas que se desea dividir la ventana de Vic para distribuir las fuentes de vídeo. Así, si, por ejemplo, establecemos un valor de 2, las fuentes de vídeo se distribuirán en la pantalla de Vic dividida en dos columnas. Esto es útil para tener mejor organizada las ventanas.
  - **Position Window** [**Valores:** *Off / Justify Left / Justify Right*, **Por defecto:** Justify Left]: Esta opción posiciona automáticamente la ventana de la aplicación Vic. Si elegimos el valor *Off* la posición será la que nosotros hayamos puesto. Si elegimos el valor *Justify Left*, colocará la ventana a la izquierda, y si elegimos el valor *Justify Right* colocará la ventana a la derecha.

Configuración a través del Gestor de Servicios (Versión Windows)

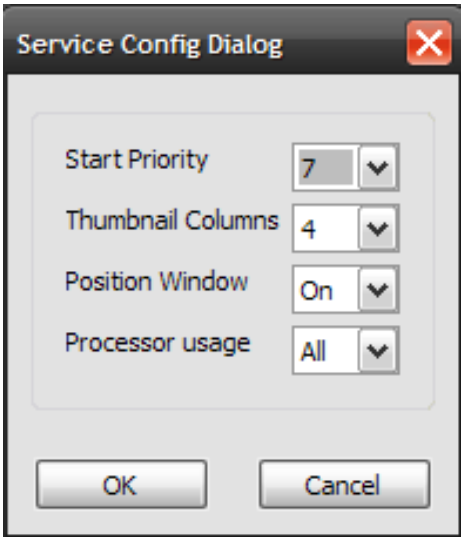
Hablamos de la versión concreta de Vic para los Sistemas Operativos Windows ya que incluye una **opción extra que es muy importante**. Para configurar Vic como visualizador a través del Gestor de Servicios, hacemos lo siguiente:

- En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



- Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Hacemos doble click en ***VideoConsumerService***, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:

  Nota: Hay que recordar que hay que tener añadido el servicio VideoConsumerService, en caso de que no aparezca en el listado, hay que añadirlo.



Podemos observar que, prácticamente aparecen las mismas opciones que en la versión de Ubuntu, excepto la última opción. Recomendamos que visite el apartado anterior (**Versión Unix**), donde se explica estas opciones comunes que hay entre ambos sistemas operativos.

La opción extra que dispone los sistemas Windows es:

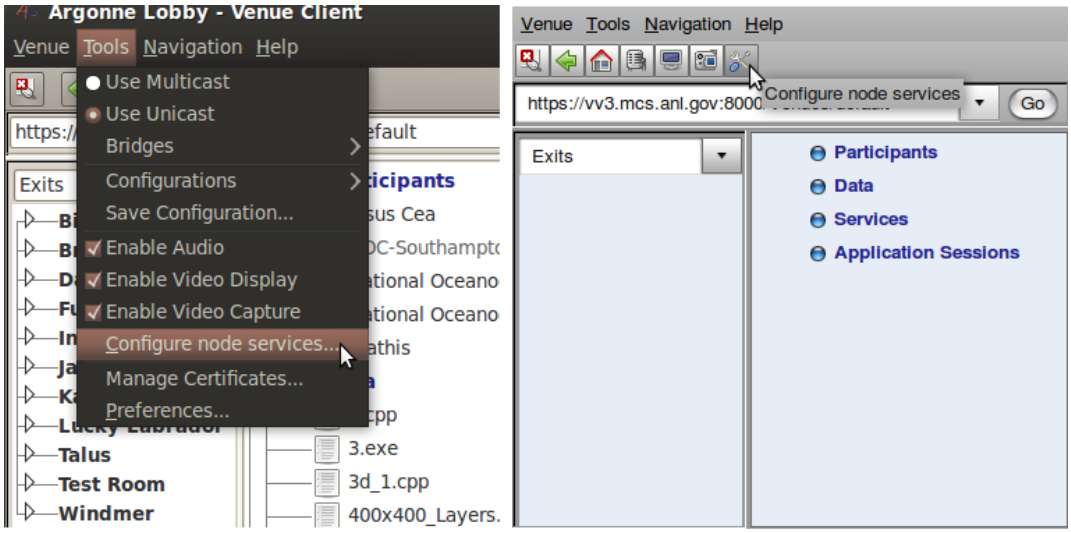
- **Processor Usage** [**Valores:** *All / 1-N* donde N es el número de núcleos de tu CPU, **Por defecto:** *All*]: Esta opción activa una función especial de Windows llamada **Establecer Afinidad**. Dicha función permite indicarle al sistema cuántos núcleos queremos que se usen para ejecutar un proceso, en nuestro caso, el proceso Vic. Así, si está en valor All, indicamos que queremos que se

usen todos los núcleos que posea la CPU, si indicamos 2 estamos diciéndole al sistema que utilice únicamente 2 núcleos de los disponibles, etc. Esta opción es **muy importante**, ya que, se ha encontrado problemas de rendimiento en sistemas multinúcleos bajo Windows y, **estableciendo este valor a 1**, mejora considerablemente el rendimiento de Vic.

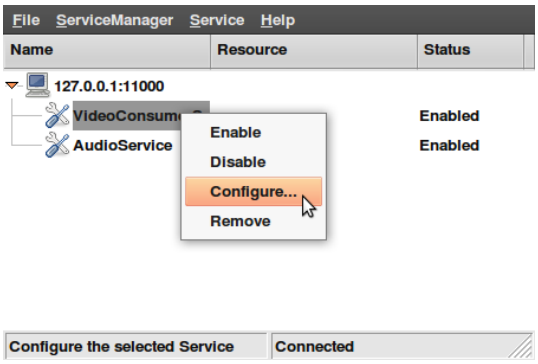
Cómo hacer que los cambios realizados surtan efecto

Como se ha comentado anteriormente, el inconveniente que tiene utilizar este método es que, una vez realizado los cambios oportunos, no surtirán efecto a no ser que recarguemos el servicio de Visualizador de Vídeo, o volvamos a entrar en la sala.

- **Volver a entrar a la sala:** Simplemente, volviendo a pulsar sobre el botón Go volveremos a entrar en la sala.
- **Recargar el servicio de Visualizador de Vídeo:**
  - En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



  - Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Seleccionamos el servicio de Visualizador de Vídeo (***VideoConsumerService***) y pulsamos el segundo botón del ratón. Le damos a ***Disabled*** para deshabilitar el servicio. Luego, repetimos el proceso para elegir, esta vez, ***Enabled***, habilitando así, de nuevo, el servicio de Visualizador de Vídeo:



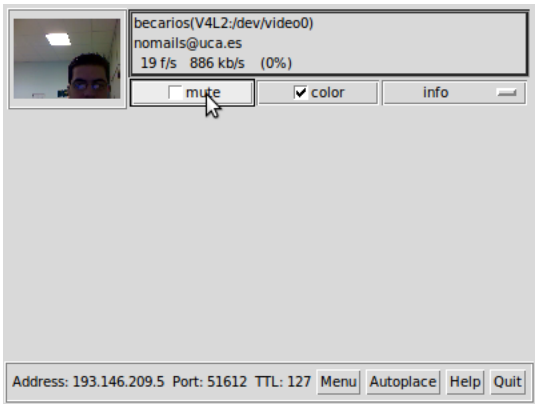
Hay que recordar que, una vez realizado los cambios oportunos, según nuestras necesidades, podemos guardarlas en un fichero de configuración en Access Grid. Esto se explicará en un próximo capítulo, ya que, dicho fichero de configuración no guarda únicamente la configuración del vídeo, sino una configuración global de nuestros servicios añadidos en Access Grid.

Configuración desde Vic

Otro de los posibles métodos de configurar la aplicación Vic es configurarla directamente desde la propia aplicación. Este método es directo ya que se está configurando desde la propia aplicación y, además, a diferencia del método anterior, los cambios que se hagan toman efecto en el mismo instante. Además ofrece muchas más opciones de configuración, por lo que da más versatilidad a la configuración del vídeo.

Sin embargo, utilizar este método tiene algunos inconvenientes. Por un lado, los cambios que hagamos en él no podremos guardarlo para futuras sesiones, por lo que tendremos que repetir el procedimiento cada vez que entremos en una sala (a no ser que guardemos o creemos un fichero de configuración, en próximos capítulos se explicará cómo).

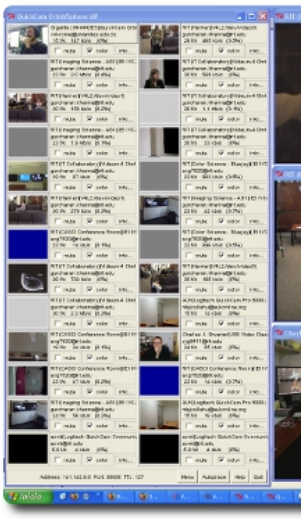
No obstante, para los usuarios avanzados que quieran experimentar y/o exprimir al máximo la configuración de audio, se detallarán todas las opciones que ofrece esta aplicación. Como ya sabe, una vez iniciada una sesión de Access Grid y, si tenemos el servicio de Visualizador de Vídeo añadido a nuestra lista de servicios (dentro del **Gestor de Servicios del Nodo**), nos debe aparecer, entre otras cosas, la aplicación Vic:



Cabe destacar que este servicio **debe ser añadido y configurado en aquella máquina que se vaya a considerar como máquina de Visualización**. Es decir, en configuraciones Multi-Nodo, donde las funciones se dividen entre varias máquinas, si, como es nuestro caso, tenemos una máquina para la visualización de vídeo y otra máquina para la captura y emisión de vídeo, la configuración de este Vic concreto dedicado a la visualización debe hacerse en la máquina de Visualización.

Esta pantalla podemos observar que se divide en dos partes:

- **Listado de Participantes Conectados:** Son las fuentes de vídeo que recibiremos en una sesión de Access Grid. Habrá tantas filas de usuarios como cámaras u otras fuentes de vídeo (escritorio, etc.) se transmitan. Aquí podemos ver un ejemplo con varias fuentes de vídeo:



- Dentro de este listado, **por cada fuente de vídeo**, tenemos las siguientes datos y opciones:

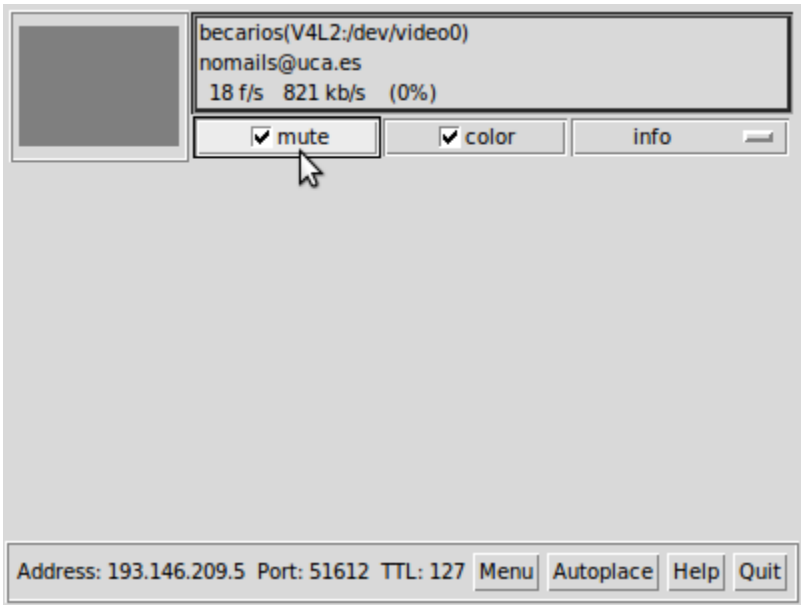
  - **Información general:** Es un panel donde se muestra el nombre del usuario que transmite la fuente de vídeo junto a su SiteId. Además, podemos ver otros datos como la tasa de frames por segundo a la que estamos recibiendo la fuente de vídeo, la tasa de trasferencia en Kb/s y el porcentaje de pérdida de paquetes.
  - **Mute**: Como su nombre indica, si está marcada, sirve para apagar una fuente de vídeo y dejar de recibirla. Desmarcándola volvemos a recibir información de la fuente de vídeo.
  - **Color:** Aunque no hemos notado ninguna diferencia marcándola y desmarcándola, supuestamente esta opción activa (si está marcada) o desactiva el color de la fuente de vídeo. Es decir, si está desmarcada deberíamos de recibir la fuente de vídeo en blanco y negro.
  - **Info:** Se trata de un menú desplegable donde aparece varias categorías, las cuales, según la que seleccionemos nos mostrará información en relación a la categoría seleccionada. Las categorías disponibles son (*depende de la versión de Vic*):
    - **Site Info:** Muestra información completa sobre el perfil de la fuente de vídeo, su SiteId, el códec al que está emitiendo el vídeo y a qué resolución, etc.
    - **RTP Stats:** Vic emite y recibe fuentes de vídeo a través de la red utilizando el protocolo RTP[5]. Desde este panel, podremos ver información acerca de dicha transmisión de datos, visualizando un conjunto de medidas, entre las cuales, podemos destacar: los kilobits, frames por segundo y el número de paquetes perdidos. Toda esta información se desglosa en tres columnas: EWA (media), Delta (número actual), Total.
    - **Decoder Stats:** Al igual que el panel anterior, muestra un conjunto de medidas desglosadas en tres columnas (EWA, Delta y Total) relacionadas con el códec de la fuente de vídeo.
    - **Decoder Control:** Dependiendo del códec seleccionado, se muestra un panel de control para configurar el códec. En caso de que no se pueda configurar aparecerá únicamente el nombre del participante al que pertenece la fuente de vídeo.
  - **Barra de Estado y Botones:** La barra de estado muestra nuestra dirección IP, puerto y Tiempo de vida de nuestra conexión. Los botones que aparecen son:

[5] `http://es.wikipedia.org/wiki/Real-time_Transport_Protocol`

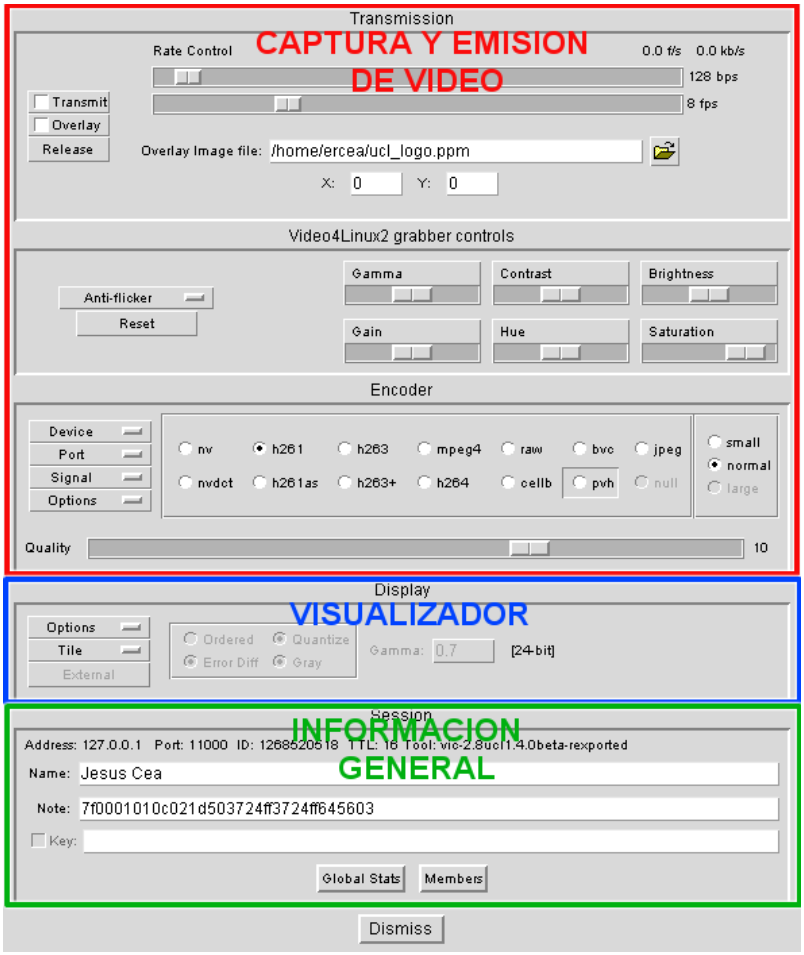
- **Menu:** Muestra el menú de configuración de Vic. Este menú se explicará en la sección Configuración avanzada de Vic como Visualizador (VideoConsumer y variantes).
- **Autoplace:** Muestra la ventana de configuración para la auto-colocación de ventanas de fuentes de vídeo que abramos con Vic.
- **Help:** Muestra una ventana de ayuda general, donde explica, de forma resumida, las funciones básicas de Vic.
- **Quit:** Cierra la aplicación Vic (y todas las fuentes de vídeo abiertas con ella).

Configuración avanzada de Vic como Visualizador (*VideoConsumer* y variantes)

En este apartado se detallará cada una de las opciones, **destinadas a la configuración del Visualizador**, que ofrece la aplicación Vic al pulsar el botón **Menu** de la interfaz principal del programa:



Cuando pulsamos sobre el botón Menu de la aplicación Vic, nos aparecerá una ventana como la que se muestra a continuación:



En dicha ventana podemos encontrar tres partes bien diferenciadas:

- **Panel de Configuración de la Captura y Transmisión de Vídeo:** Que representa la zona de recuadro rojo. Son opciones cuya configuración afecta a la captura y emisión de vídeo
- **Panel de Configuración de la Visualización de Vídeo:** Que representa la zona de recuadro azul. Son opciones cuya configuración afecta a la Visualización (recepción) de vídeo.
- **Panel de Información General:** Que representa la zona de recuadro verde. Muestra información general sobre el nuestro perfil, nuestro SiteId, además de dos botones:
  - **Globat Stats:** Donde se muestra unas estadísticas globales sobre la trasferencia de los paquetes de vídeo (paquetes perdidos, etc.)
  - **Members**: Donde se muestra un listado de los participantes que están conectados actualmente a la sesión.

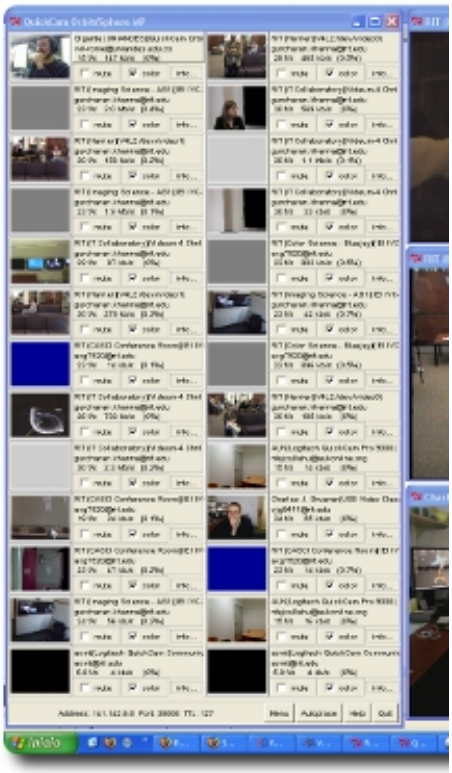
Debido a que esta sección se dedica **exclusivamente a la configuración de Vic como Visualizador**, únicamente explicaremos el conjunto de opciones destinadas a dicha funcionalidad, es decir, al conjunto de opciones que están en el recuadro azul.

En dicho panel podemos encontrar las siguientes opciones:

- **Options:** Conjunto de opciones:

- **Mute New Sources:** Si esta opción está activa se silenciarán (desactivarán) las nuevas fuentes de vídeo que se conecten a la sesión.
- **Use Hardware Decode:** Si esta opción está activa se utilizará la aceleración por hardware para decodificar las fuentes de vídeo entrantes.

- **Tile** [**Valores:** *Single / Double / Triple / Quad*, **Por defecto:** *Single*]: Divide la visualización de las fuentes de vídeo en Vic en tantas columnas como se indique. Sirve para tener una mejor organización de las fuentes de vídeo a visualizar en Vic. En esta imagen, por ejemplo, se puede apreciar que el valor de *Tile* es **Double**:



Para finalizar esta sección, es muy recomendable que lea la sección ***Problemas de rendimiento del equipo debido a Vic*** para solucionar posibles problemas de rendimiento en el equipo debido a la aplicación Vic como Visualizador.

Configuración de Vic como Productor (*VideoProducer* y variantes)

La aplicación Vic (*Video Conference Tool*[6]) viene integrada en el sistema Access Grid. Cuando entramos en una sesión de Access Grid, si tenemos añadido, o queremos añadir, un servicio de vídeo, bien sea productor o visualizador, se ejecutará automáticamente para gestionar el vídeo, tanto la emisión, como la recepción, dependiendo del servicio añadido. En esta sección se explicará cómo configurar Vic como **Productor** (*VideoProducer* y variantes).

Podemos configurar Vic de tres formas distintas:

[6] `http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/index.html`

- A través del Gestor de Servicios (*Tools* $-$ $>$ *Configure node services*), que será la recomendada, ya que, los cambios que hagamos desde aquí, podrán ser guardados en un fichero de configuración para no tener que repetir el procedimiento. Además, desde aquí podemos configurar los servicios que estén alojados en otra máquina (a través de ServiceManager, ver *Añadir Servicios situados en otra máquina*), y, como antes, también podremos guardar la configuración en un fichero de configuración.

- Desde el propio Vic: Se configura directamente desde la propia aplicación. Recomendado para configuraciones concretas, ya que los cambios que se haga no podrán ser guardados en un fichero de configuración. Sin embargo, al tratarse de la propia aplicación, permite una configuración más avanzada que la que ofrece el Gestor de Servicios.

- Crear/Editar un fichero de configuración: Access Grid permite guardar ficheros de configuración, donde se almacenan las configuraciones de los servicios que usemos (audio, vídeo, etc.) en nuestra sesión de Access Grid. Si uno de los servicios que usamos es un servicio de vídeo (ya sea visualizador o productor), podemos editar nuestro fichero para configurar aquellas opciones que ofrece Vic pero no están disponibles desde el Gestor de Servicios. En capítulos posteriores se explicará con detalle cómo editar o crear un fichero de configuración personalizado.

Como en esta sección se explicará Vic como Productor, hay que asegurarse de que hemos añadido el servicio VideoProducerService o alguna de sus variantes (*VideoService*, *VideoProducerServiceH264*, etc.). Para añadir el servicio visite:

- **Añadir Servicios en Windows XP:** Si su sistema es un sistema Windows

- **Añadir Servicios en Ubuntu:** Si su sistema es Ubuntu o cualquier otro sistema Unix

Por otro lado, hay que destacar que **hay que añadir un servicio por cada fuente de vídeo que se quiera transmitir**. Si, por ejemplo, tenemos tres cámaras, tendremos que añadir 3 servicios VideoProducerService (o sus variantes). Si, por ejemplo, tenemos 3 cámaras y, además, queremos enviar, como fuente de vídeo, nuestro escritorio, tendremos que añadir 4 servicios VideoProducerService (o sus variantes). La configuración que se realice únicamente afecta al servicio de vídeo elegido. Si queremos configurar todas las fuentes de vídeo, **hay que repetir el proceso de configuración para cada una de ellas**.

Configuración a través del Gestor de Servicios

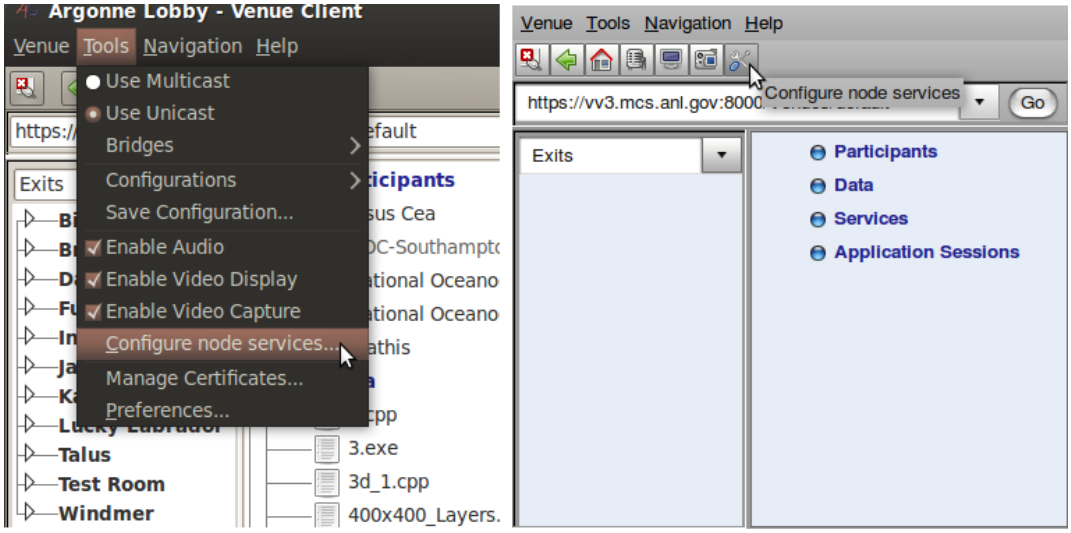
Uno de los métodos para poder configurar Vic es a través del **Gestor de Servicios del Nodo** de Access Grid. Vic es una aplicación independiente que ha sido integrada en el sistema Access Grid, así que, desde este método, nos aparecerá una interfaz sencilla con las opciones que se han considerado relevantes para el buen funcionamiento del Vídeo en Access Grid.

Sin embargo, utilizar este método de configuración tiene el inconveniente de que los cambios que hagamos en la configuración no surtirán efecto a no ser que, o bien deshabilitemos y habilitemos el servicio de Video como **Productor** (*VideoProducer* y variantes), o bien, volviendo a entrar en la Sala.

Por similitud de las diferentes variantes existentes del Servicio de Visualización de Vídeo (*VideoProducerService*, *VideoProducerServiceH264*,etc), nos basaremos en el servicio estándar que ofrece Access Grid, éste es el servicio VideoConsumerService.

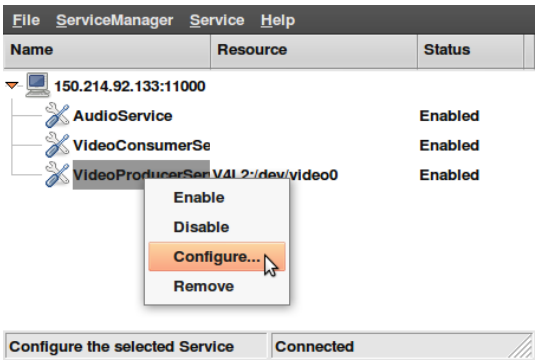
Para configurar Vic como Productor a través del Gestor de Servicios, hacemos lo siguiente:

- En el Cliente de Sala (VenueClient), nos dirigimos a "Tools − > Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:

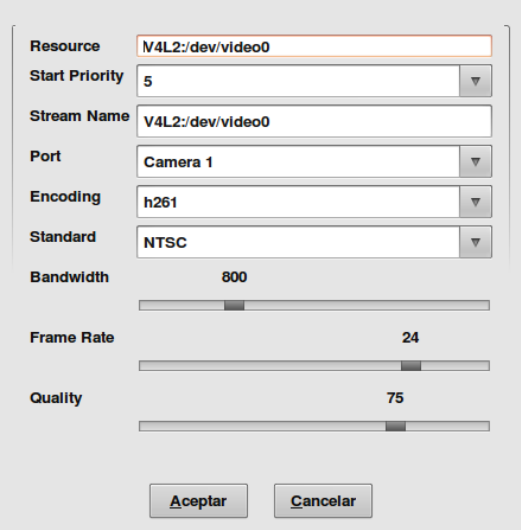


- Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Hacemos doble click en **VideoProducerService**, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:

  Nota: Hay que recordar que hay que tener añadido el servicio VideoProducerService, en caso de que no aparezca en el listado, hay que añadirlo.



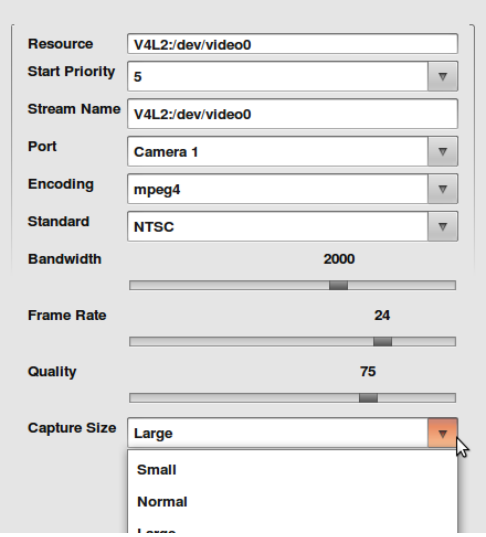
- Nos aparecerá la pantalla de configuración del Vic, como Productor:



- Las opciones que podemos encontrar son:

  - **Resource:** No permite modificar. Muestra la fuente de vídeo que se está utilizando como recurso (V4L2:/dev/video0, video1, etc.)
  - **Start Priority** [**Valores:** *1 al 10*, **Por defecto:** *7*]: Esta opción nos permite variar la prioridad de este proceso, a mayor prioridad, más uso de CPU se le asigna.
  - **Stream Name:** Permite cambiar el identificador del flujo de vídeo que se transmite.
  - **Port** [**Valores:** *Camera1-CameraN* (tantas como cámaras/capturadoras haya instalada en el equipo, **Por defecto:** *Camera1*]: Esta opción permite elegir el dispositivo de vídeo que queremos utilizar como fuente de vídeo a transmitir.
  - **Encoding** [**Valores:** (*Depende del servicio de vídeo utilizado) H.261 / H.264 / MPEG-4 / ...* , **Por defecto:** *Depende del servicio de vídeo utilizado*]: Esta opción cambia el códec de vídeo a utilizar para la codificación y emisión del vídeo. Si el servicio utilizado es el **VideoService** o **VideoProducerService** únicamente nos dejará elegir el códec H.261. Sin embargo, si utilizamos el servicio **VideoProducerServiceH264**, podremos elegir entre MPEG-4 / H.264 y H.261a. Si añadimos algún otro servicio de vídeo específico, dependerá de los códecs que proporcione dicho servicio (por ejemplo el servicio JPEGVideoService y JPEGVideoProducerService permite elegir el códec Jpeg como códec de vídeo).
  - **Standard:** [**Valores:** *NTSC / PAL*, **Por defecto:** *NTSC*]: Esta opción permite elegir el sistema de codificación de vídeo entre PAL o NTSC. Por defecto está el valor NTSC, sin embargo, en la mayoría de los sistemas europeos deberemos de elegir el sistema **PAL**.
  - **Bandwith** [**Valores:** *0-3072*, **Por defecto:** *2000*]: Esta opción permite establecer el bitrate, es decir, el número de bits que se transmiten por unidad de tiempo a la red. Este valor afecta a la calidad de la red, por lo que es conveniente modificar este valor para adecuarlo a las características de su red (al ancho de banda del que dispone). Desde Access Grid se aconseja un valor promedio de **800 kbps**.
  - **Frame Rate** [**Valores:** *1-30*, **Por defecto:** *24*]: Esta opción permite establecer la tasa de imágenes por segundo(medido en fps) al que queremos transmitir la fuente de vídeo. Se recomienda transmitir a **25 fps**.
  - **Quality** [**Valores:** *1-100*, **Por defecto:** *75*]: Esta opción permite establecer la calidad de la fuente de vídeo. A más a la derecha, mayor valor y, por tanto, **mayor calidad de vídeo pero, a su vez, mayor consumo de recursos**.

- **Otras opciones:** Dependiendo del servicio de vídeo como Productor pueden aparecer otras opciones. Por ejemplo, **VideoProducerServiceH264** tiene una opción:



  - **Capture Size:** [**Valores:** *Large / Medium / Small*, **Por defecto:** *Large*]: Esta opción permite cambiar la resolución de vídeo. Si la opción elegida es **Large**, la fuente de vídeo será enviada a una resolución de 720x576. Si el tamaño elegido es **Medium**, se enviará a una resolución de 320x240. Por último, la opción **Small** envía la fuente de vídeo a una resolución de 180x150. La resolución de vídeo también afecta al rendimiento del equipo. Por lo que, si en una sesión va a haber muchas fuentes de vídeo, se recomienda utilizar el tamaño **Medium**.

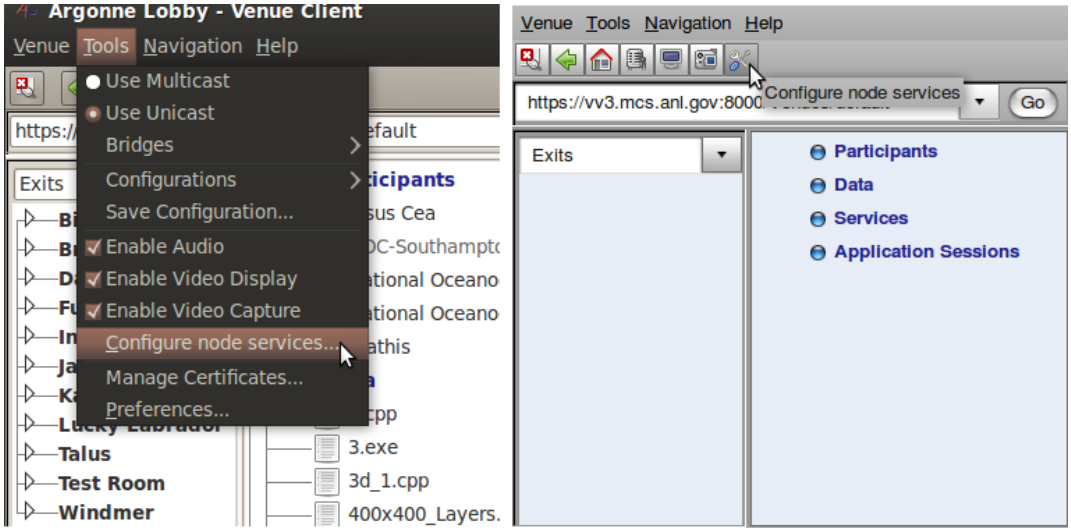
Cómo hacer que los cambios realizados surtan efecto

Como se ha comentado anteriormente, el inconveniente que tiene utilizar este método es que, una vez realizado los cambios oportunos, no surtirán efecto a no ser que recarguemos el servicio de Visualizador de Vídeo, o volvamos a entrar en la sala.

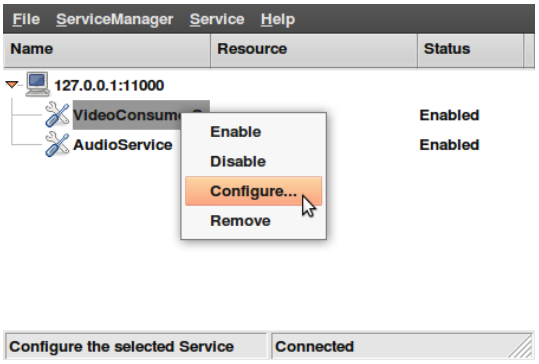
- **Volver a entrar a la sala:** Simplemente, volviendo a pulsar sobre el botón Go volveremos a entrar en la sala.

- **Recargar el servicio de Visualizador de Vídeo:**

  - En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



- Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Seleccionamos el servicio de Visualizador de Vídeo (***VideoProducerService***) y pulsamos el segundo botón del ratón. Le damos a ***Disabled*** para deshabilitar el servicio. Luego, repetimos el proceso para elegir, esta vez, ***Enabled***, habilitando así, de nuevo, el servicio de Visualizador de Vídeo:



Hay que recordar que, una vez realizado los cambios oportunos, según nuestras necesidades, podemos guardarlas en un fichero de configuración en Access Grid. Esto se explicará en un próximo capítulo, ya que, dicho fichero de configuración no guarda únicamente la configuración del vídeo, sino una configuración global de nuestros servicios añadidos en Access Grid.

Configuración desde Vic

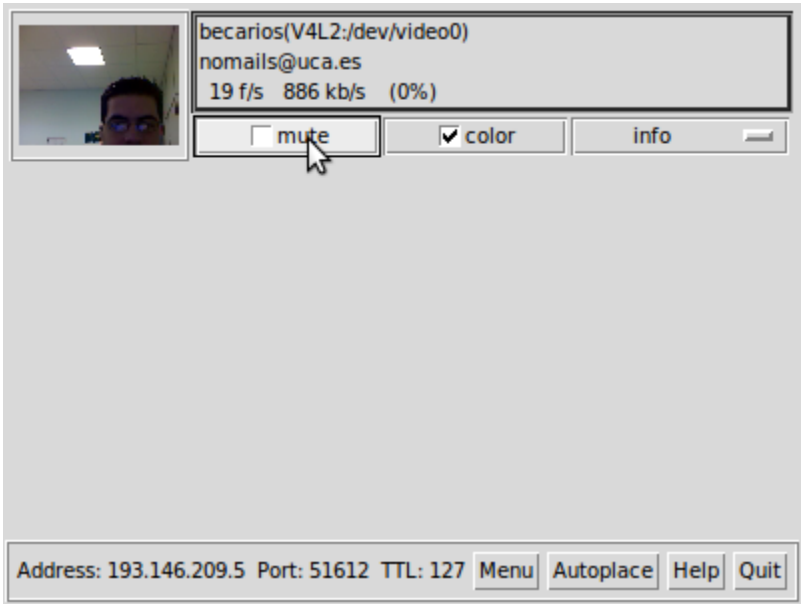
Otro de los posibles métodos de configurar la aplicación Vic es configurarla directamente desde la propia aplicación. Este método es directo ya que se está configurando desde la propia aplicación y, además, a diferencia del método anterior, los cambios que se hagan toman efecto en el mismo instante. Además ofrece muchas más opciones de configuración, por lo que da más versatilidad a la configuración del vídeo.

Sin embargo, utilizar este método tiene algunos inconvenientes. Por un lado, los cambios que hagamos en él no podremos guardarlo para futuras sesiones, por lo que tendremos que repetir el procedimiento cada vez que entremos en una sala (a no ser que guardemos o creemos un fichero de configu-

ración, en próximos capítulos se explicará cómo).

No obstante, para los usuarios avanzados que quieran experimentar y/o exprimir al máximo la configuración de audio, se detallarán todas las opciones que ofrece esta aplicación.

Como ya sabe, una vez iniciada una sesión de Access Grid y, si tenemos el servicio de Productor de Vídeo añadido a nuestra lista de servicios (dentro del Gestor de Servicios del Nodo), nos debe aparecer, entre otras cosas, la aplicación Vic:



Cabe destacar que este servicio **debe ser añadido y configurado en aquella máquina que se vaya a considerar como máquina de Captura y Emisión de Vídeo**. Es decir, en configuraciones Multi-Nodo, donde las funciones se dividen entre varias máquinas, si, como es nuestro caso, tenemos una máquina para la visualización de vídeo y otra máquina para la captura y emisión de vídeo, la configuración de este Vic concreto dedicado a la captura y emisión de las fuentes de vídeo debe hacerse en la máquina de Captura y Emisión.

Por otro lado, debemos recordar que **hay que añadir un servicio por cada fuente de vídeo que se quiera transmitir**. Si, por ejemplo, tenemos tres cámaras, tendremos que añadir 3 servicios *VideoProducerService* (o sus variantes). Si, por ejemplo, tenemos 3 cámaras y, además, queremos enviar, como fuente de vídeo, nuestro escritorio, tendremos que añadir 4 servicios *VideoProducerService* (o sus variantes). La configuración que se realice únicamente afecta al servicio de vídeo elegido. Si queremos configurar todas las fuentes de vídeo, **hay que repetir el proceso de configuración para cada una de ellas**.

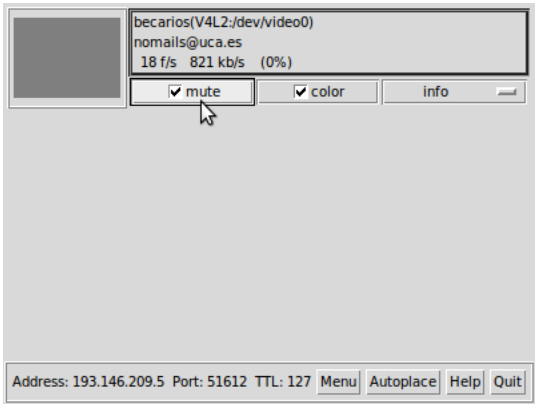
Configuración desde un fichero de configuración

Como se ha comentado varias veces en secciones anteriores, puede guardar qué servicios quiere que se ejecuten al entrar en una sala, así como la configuración de cada uno de ellos. De esta forma, se ahorra tener que repetir el mismo procedimiento cada vez que entre en una sala. A continuación se explicará los pasos a seguir para guardar una configuración para luego cargarla.

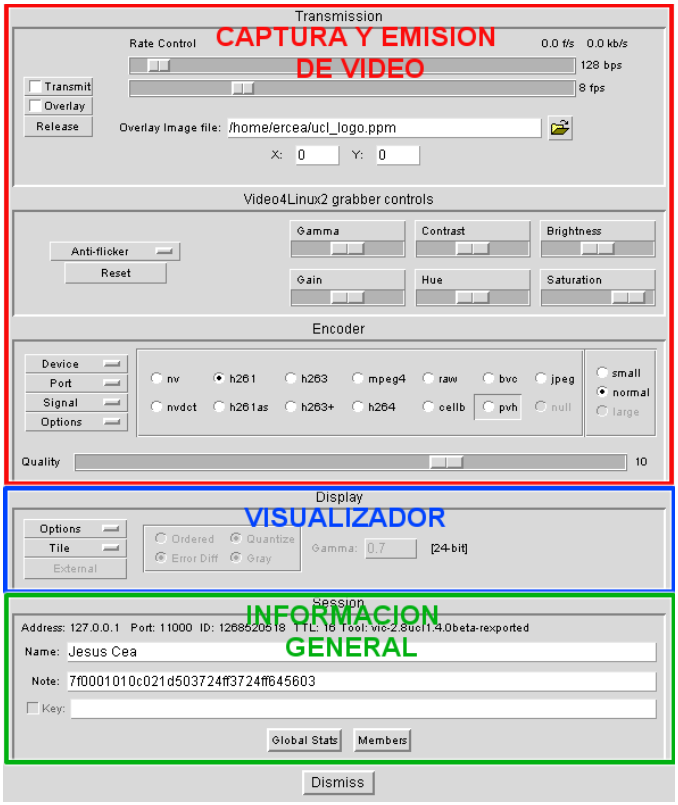
Por otro lado, estas configuraciones almacenadas, lo que realmente hace Access Grid es almacenar cada configuración en un fichero, almacenado en una carpeta del Cliente, luego, para cargar la configuración, lee ese fichero para aplicar los cambios. Así que, para usuarios más experimentados o para aquellos que tengan algún problema en almacenar la configuración, se explicará brevemente cómo crear un fichero de configuración personalizado desde un editor de textos.

Configuración avanzada de Vic como Productor (*VideoProducer* y variantes)

En este apartado se detallará cada una de las opciones, **destinadas a la configuración de Captura y Emisión de Vídeo**, que ofrece la aplicación Vic al pulsar el botón **Menú** de la interfaz principal del programa:



Cuando pulsamos sobre el botón Menu de la aplicación Vic, nos aparecerá una ventana como la que se muestra a continuación:



En dicha ventana podemos encontrar tres partes bien diferenciadas:

- **Panel de Configuración de la Captura y Transmisión de Vídeo:** Que representa la zona de recuadro rojo. Son opciones cuya configuración afecta a la captura y emisión de vídeo.
- **Panel de Configuración de la Visualización de Vídeo:** Que representa la zona de recuadro azul. Son opciones cuya configuración afecta a la Visualización (recepción) de vídeo.
- **Panel de Información General:** Que representa la zona de recuadro verde. Muestra información general sobre el nuestro perfil, nuestro SiteId, además de dos botones:
  - **Globat Stats:** Donde se muestra unas estadísticas globales sobre la trasferencia de los paquetes de vídeo (paquetes perdidos, etc.)
  - **Members:** Donde se muestra un listado de los participantes que están conectados actualmente a la sesión.

Debido a que esta sección se dedica exclusivamente a la configuración de Vic como Capturador y Emisor de Vídeo, únicamente explicaremos el conjunto de opciones destinadas a dicha funcionalidad, es decir, al conjunto de opciones que están en el recuadro rojo.

En dicho panel podemos encontrar las siguientes opciones:

- **Transmit:** Si se marca esta opción la aplicación Vic comenzará a enviar nuestra fuente de vídeo a la red, es decir, transmitirá nuestra fuente de vídeo.
- **Rate Control:** Podemos encontrar dos valores a configurar:

- **Bandwith** [**Valores:** *1-3072*, **Por defecto:** *Depende del servicio de vídeo utilizado*]: Esta opción permite establecer el bitrate, es decir, el número de bits que se transmiten por unidad de tiempo a la red. Este valor afecta a la calidad de la red, por lo que es conveniente modificar este valor para adecuarlo a las características de su red (al ancho de banda del que dispone). Desde Access Grid se aconseja un valor promedio de **800 kbps**.
- **Frame Rate** [**Valores:** *1-30*, **Por defecto:** *24*]: Esta opción permite establecer la tasa de imágenes por segundo (medido en fps) al que queremos transmitir la fuente de vídeo. Se recomienda transmitir a **25 fps**.

- **Overlay:** Si se marca esta opción activaremos el efecto *Overlay*. Este efecto superpone una imagen en formato .ppm a nuestra emisión de vídeo. Puede ser útil para aquellas ocasiones en las que no queremos que vean lo que estamos transmitiendo y ponemos una imagen fija en espera. También puede ser útil para testeo de conexiones y configuración.

- **Overlay Image file:** Indicamos la ruta de la imagen en formato .ppm que queremos que se muestre al activar el efecto Overlay. Podemos indicar la ruta también usando el botón de la carpeta que se encuentra al lado de la ruta.

- **X e Y** [**Valores:** *0-Inf*, **Por defecto:** *0*]: Son coordenadas para indicar a Vic dónde queremos colocar la imagen, en formato .ppm, que usaremos como Overlay. Una vez escrito el valor, **hay que pulsar la tecla Intro** para que surta efecto.

Dentro del panel *Video4Linux2 Grabber Controls* encontraremos diferentes opciones para configurar la imagen de vídeo:

- **Anti-Flicker** [**Valores:** *disabled / 50Hz / 60 Hz*, **Por defecto:** *disabled*]: Esta opción permite corregir un defecto llamado flicker o parpadeo que se suele dar en algunas capturadoras de vídeo. Dicho efecto produce una especie de parpadeo que se produce debido a un fenómeno de variación de la intensidad luminosa que afecta la visión humana, principalmente en el rango de frecuencias de 0 a 25 Hz. Por defecto viene desactivado, sin embargo, si nota en su vídeo captado dicho efecto de parpadeo debería de activarlo, bien al valor de 50Hz, o bien a 60Hz, dependiendo de las características de su capturadora.

- **Gamma:** Mueva la barra deslizante adjunta para ajustar el valor de **Gamma** deseado.

- **Contrast:** Mueva la barra deslizante adjunta para ajustar el valor de **Contraste** deseado.

- **Brightness:** Mueva la barra deslizante adjunta para ajustar el valor de **Brillo** deseado.

- **Gain:** Mueva la barra deslizante adjunta para ajustar el valor de **Ganancia** deseado.

- **Hue:** Mueva la barra deslizante adjunta para ajustar el valor de Matiz de **Color** deseado.

- **Saturation:** Mueva la barra deslizante adjunta para ajustar el valor de **Saturación** deseado.

- **Botón Reset:** Restaura a valores por defecto de los parámetros Anti-Flicker y los relacionados con la configuración de imagen (Gamma, Contrate, etc.).

Dentro del panel **Encoder** podemos encontrar las siguientes opciones:

- **Device:** Selecciona una fuente de imagen que se utilizará para transmitir vídeo de una lista de todas las fuentes de imágenes (capturadoras, X11, etc.) instaladas en el equipo.
- **Port:** Selecciona el puerto por el que la fuente de vídeo, seleccionada anteriormente, utilizará para emitir vídeo.
- **Signal:** Selecciona el tipo de señal que se desea transmitir (PAL, NTSC, SECAM, etc., depende de la fuente de vídeo instalada y seleccionada).
- **Options:**
  - **Sending Slides:** Por determinar
  - **Use JPEG for H.261:** Activando esta opción indicamos a la aplicación Vic que, en el caso de que seleccionemos como códec de vídeo a utilizar el códec H.261, utilice el códec JPEGVídeo.
  - **Use Hardware Encode:** Activando esta opción activamos la aceleración or hardware para la captura y emisión de vídeo.
- **Listado de Códecs**[**Valores:** (Depende del servicio de vídeo utilizado) *H.261 / H.264 / MPEG-4 / ...* , **Por defecto:** *Depende del servicio de vídeo utilizado*]: Esta opción cambia el códec de vídeo a utilizar para la codificación y emisión del vídeo. Si el servicio utilizado es el **VideoService** o **VideoProducerService** únicamente nos dejará elegir el códec H.261 y **otros códecs de peor calidad** (nv, nvdtc, cellb, raw, etc.). Sin embargo, si utilizamos el servicio **VideoProducerServiceH264**, podremos elegir entre MPEG-4 / H.264 y H.261, además de los otros códecs de peor calidad. Si añadimos algún otro servicio de vídeo específico, dependerá de los códecs que proporcione dicho servicio (por ejemplo el servicio JPEGVideoService y JPEGVideoProducerService permite elegir el códec Jpeg como códec de vídeo).
- **Small/Normal/Large:** Esta opción permite cambiar la resolución de vídeo. Si la opción elegida es **Large**, la fuente de vídeo será enviada a una resolución de 720x576. Si el tamaño elegido es **Medium**, se enviará a una resolución de 320x240. Por último, la opción **Small** envía la fuente de vídeo a una resolución de 180x150. La resolución de vídeo también afecta al rendimiento del equipo. Por lo que, si en una sesión va a haber muchas fuentes de vídeo, se recomienda utilizar el tamaño **Medium**.
- **Quality** [**Valores:** *29-1*, **Por defecto:** *10*]: Esta opción permite establecer la calidad de la fuente de vídeo. A más a la derecha, mayor valor y, por tanto, **mayor calidad de vídeo pero, a su vez, mayor consumo de recursos**.

Aplicar los cambios

La mayoría de cambios que se realicen pueden surtir efecto al instante. Sin embargo, hay otros en los que es necesario reiniciar la transmisión. Así pues se recomienda, para que cualquier tipo de cambio surta efecto, se hagan los siguientes pasos:

- Pulse el botón **Release**, esto, cancelará la transmisión, además del Overlay.

- Pulse el botón **Transmit** para volver a transmitir el vídeo con los cambios aplicados. Si desea enviar, además, el efecto Overlay deberá volver activarlo pulsando el botón **Overlay**.

Para salir de este panel pulse el botón **Dismiss** o cierre la ventana.

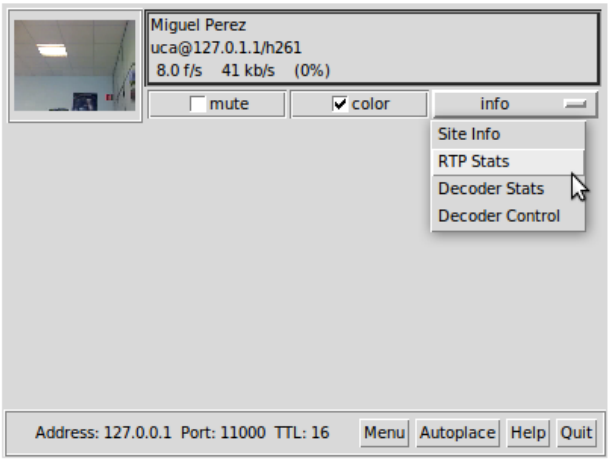
A.6.2. Problemas de rendimiento del equipo debido a Vic

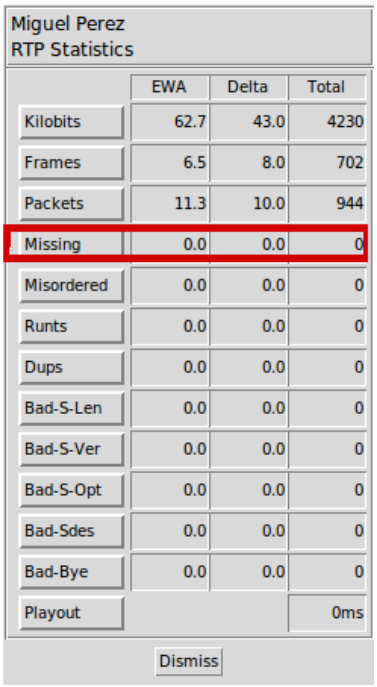
Después de varias pruebas y sesiones, se ha detectado un problema de rendimiento de la aplicación Vic, en aquellos PCs que se utilicen como **Visualizador** (recepción de las fuentes de vídeo). Este problema de rendimiento afecta tanto a Ubuntu como a Windows. Sin embargo, en Windows el problema es más grave.

Efecto

Dicho problema produce, **en el PC de Visualización**, una serie de pixelaciones prolongadas, dejando una especie de ”estela” de lo que se está recibiendo. Este efecto se hace más notorio en aquellas fuentes de vídeo donde exista algún movimiento. Además, podemos ”forzar” esta pixelación a la hora de arrastrar las ventanas para distribuirlas entre los proyectores instalados.

Una forma de cuantificar esa pixelación producida es visualizando el número de paquetes perdidos. Para ello, vaya a la aplicación Vic, pulse el botón **Info** y luego seleccione **RTP Stats**. Observando la fila de **Missing** aparecen la media de paquetes perdidos, el número actual y el número total de paquetes perdidos.





Causas de la Pixelación

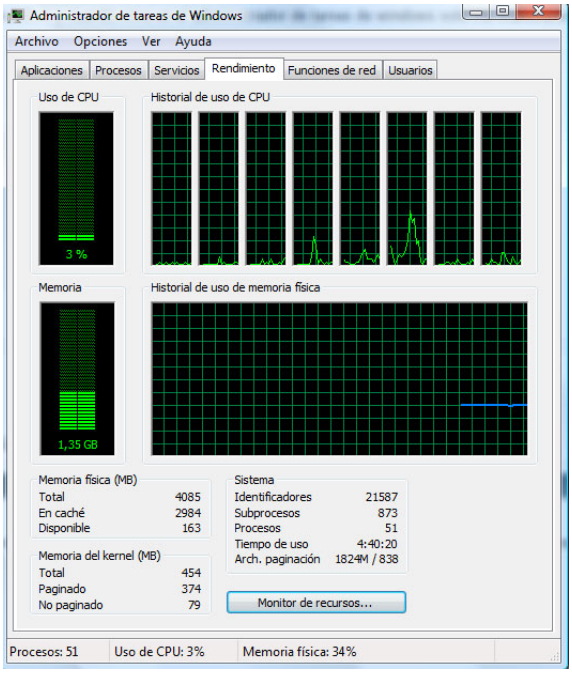
El motivo principal de que se produzca esta pixelación es la sobrecarga de la aplicación Vic, en el que dicho proceso (como **Visualizador**) llega a consumir prácticamente todo el procesador por varios motivos:

- Mala Gestión en los procesadores multinúcleos, dedicándole uno o dos núcleos la carga de trabajo mientras el resto no hacen nada (**Sólo en S.O.'s Windows**).
- Códecs de vídeo que consumen mucha CPU (MPEG-4 / H.264 etc.) y utilizando un número considerado de cámaras.
- Alta resolución de pantalla (escritorio extendido).

Problemas con procesadores multinúcleos (Sólo en S.O.'s Windows)

Este problema de pixelación se hace más notorio en sistemas Windows. Por ejemplo, en un sistema Unix, con una CPU QuadCore a unos 2'6Ghz, podemos abrir hasta cuatro ventanas de vídeo procedentes de la aplicación Vic, en formato **MPEG-4**. En cambio, con el mismo procesador, en una máquina Windows, con sólo una ventana en MPEG-4, simplemente con arrastrarla ya produce pixelaciones.

Esto es debido a una mala gestión de los sistemas Windows de los procesadores multinúcleos, ya que, si le damos al **Administrador de Tareas** (*Ctrl* + *Alt* + *Supr*), podremos observar cómo, en general, el sistema indica que se está usando un 40 % aproximadamente de CPU. Sin embargo, si nos dirigimos a la pestaña **Rendimiento**, nos aparecerá, entre otras cosas, cada uno de los núcleos que dispone nuestra CPU, así como la carga de trabajo que tiene cada uno de los núcleos, en forma de gráfica:



Observaremos cómo, al ejecutar la aplicación Vic y empezar las pixelaciones (o produciéndolas nosotros mismos al arrastrar una ventana), únicamente se dispara el uso de CPU en uno o dos núcleos, mientras que el resto apenas se aprovechan.

Solución

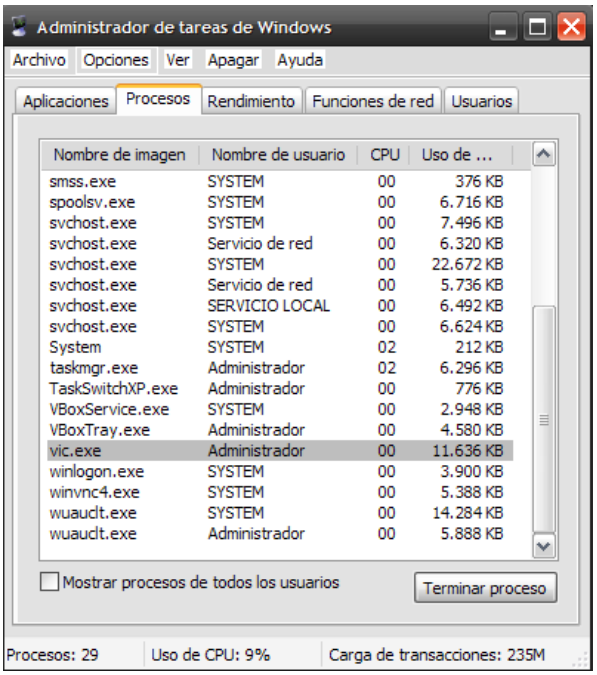
Como se ha comentado anteriormente, las causas de las pixelaciones, aunque se resume a una sobrecarga de CPU debido a la aplicación Vic, éste se sobrecarga por varios motivos. A continuación se detalla una solución a cada causa comentada:

Solución para procesadores multinúcleos (Sólo S.O.'s Windows)

Este es el problema más importante que hay que resolver si el visualizador está instalado en una máquina Windows, si usted utiliza una máquina Linux no siga estos pasos y vaya al siguiente punto.

La solución a este problema está en asignar la carga de procesador del proceso Vic a un sólo núcleo. Para ello, existen dos formas, una propia del sistema y otra, a través de Access Grid:

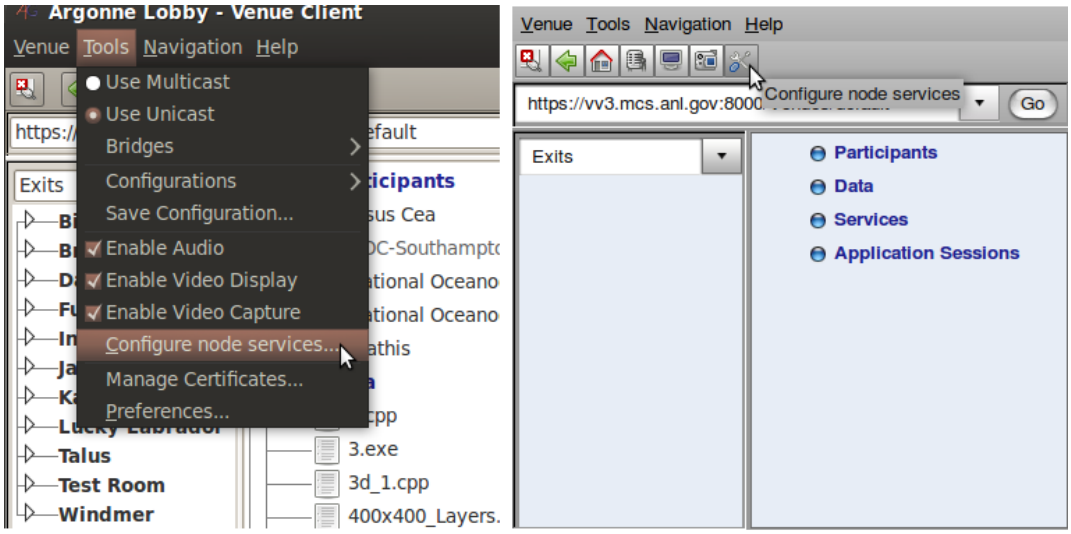
- **Desde Windows:** Ejecute el **Administrador de Tareas** (*Ctrl* + *Alt* + *Supr*) y diríjase a la pestaña **Procesos**, donde le aparecerá todos los procesos que está ejecutando actualmente el sistema Windows. Busque el proceso *Vic.exe* (puede facilitarle la búsqueda ordenando la lista por nombre) y pulse el segundo botón del ratón:



- Seleccionar **Establecer Afinidad**, nos aparecerá una ventana en el que se lista tantas CPUs como núcleos tenga nuestro sistema. Podemos elegir tantos núcleos como tengamos, o uno, o dos, etc. También podemos indicarle precisamente cuál de nuestros núcleos queremos que ejecuten la aplicación. En nuestro caso, tenemos que elegir únicamente uno (cualquiera de los que tenga nuestro sistema):

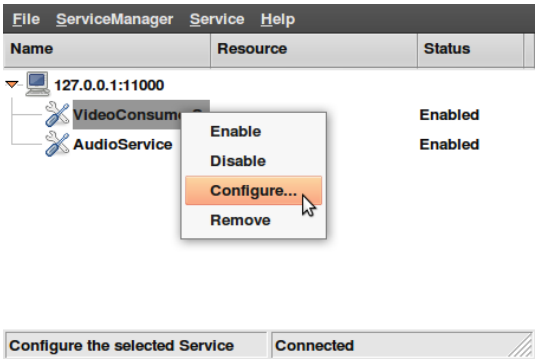
Esta solución tiene el inconveniente de que, cada vez que ejecutemos la aplicación Vic, bien sea arrancando el sistema Access Grid de nuevo o incluso al entrar de nuevo en otra sala (ya que recarga la aplicación Vic), hay que realizar estos pasos, ya que no se guarda la afinidad en el proceso Vic. Por lo que no parece una solución práctica.

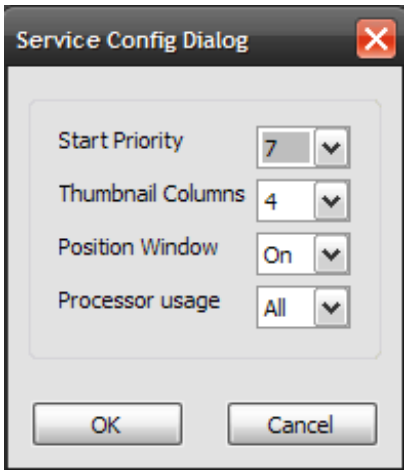
- **Desde Access Grid:** Access Grid, parece ser que ya conocían este problema, y ofrecen una solución mucho mejor y más automatizado. Recordemos que en la sección *Configuración de Vic como Visualizador (VideoConsumer y variantes)* hablábamos de dos formas de configurar Vic: desde el Gestor de Servicios de Access Grid o desde la propia aplicación. Para solucionar este problema utilizaremos el **Gestor de Servicios de Access Grid**. Para ello:
  - En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



  - Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Hacemos doble click en **VideoConsumerService**, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:

Nota: Hay que recordar que hay que tener añadido el servicio VideoConsumerService, en caso de que no aparezca en el listado, hay que añadirlo.





- En la opción **Processor Usage**, cambiamos el valor *All* por el valor **1**. Pulsamos el botón *Ok*.
- Recuerde que para que los cambios surtan efecto hay que, o bien recargar el servicio, o bien entrar de nuevo en la sala.

Esta solución es la más recomendada, ya que, una vez hagamos estos cambios, podemos guardar la configuración del Visualizador. Así, cuando entremos en otra sala, o cuando volvamos a ejecutar Access Grid, no hará falta repetir estos pasos ya que están almacenados en nuestro fichero de configuración (tendremos que cargar nuestra configuración si no es la configuración por defecto).

Solución a los códecs de vídeo y número de ventanas

El códec utilizado, la resolución de vídeo, el número de ventanas y el tamaño de éstas repercuten en el rendimiento del sistema. Dependiendo del códec utilizado, podremos utilizar un número determinado de ventanas sin que éstas se pixelen, pero, además, dependiendo del tamaño de estas ventanas podremos aumentar más aún el número de ventanas.

Por ejemplo, si utilizamos el códec MPEG-4, con resolución **Large** y queremos que las ventanas sean lo más grande posibles (pulsando la tecla **L**), únicamente podremos utilizar 3 ventanas como mucho. En cambio, si queremos ventanas de tamaño medio, podremos aumentar el número de éstas.

Por otro lado, si utilizamos en lugar del códec MPEG-4, el códec H.261, podremos poner hasta 8 ventanas de tamaño grande (pulsando la tecla **L**), y, evidentemente, podremos aumentar el número de ventanas si reducimos el tamaño de éstas a **Normal**.

Por lo tanto, como puede comprobar, se tratar de ”jugar” con estos 4 parámetros: códec a utilizar, resolución de vídeo, el número de ventanas y el tamaño de éstas.

Por último, hay que aclarar que a la hora de **arrastrar las ventanas para distribuirlas** a lo largo del escritorio (ya sea extendido o no) puede que se **produzcan pixelaciones**, ya que, dicho arrastre aumenta la carga de procesador del sistema. Sin embargo, una vez haya colocado las ventanas donde desea, al cabo de unos segundos, se llega a estabilizar el sistema y desaparecer las pixelaciones.

Resolución de pantalla

Por último, si dispone de una resolución muy alta de pantalla puede afectar negativamente al rendimiento del sistema si tenemos varias ventanas de vídeo ejecutándose (sobre todo si su tamaño es Grande). Por lo que puede considerar como un parámetro adicional la resolución de pantalla, en combinación a los parámetros anteriores mencionados (códec, resolución, tamaño y número de ventanas). Reduciendo la resolución de pantalla se consigue un mejor rendimiento del sistema y, por tanto, puede aumentar el número de ventanas de vídeo de Vic, aunque también, al tener una resolución más pequeña, dispone de menos espacio ”físico” para distribuir estas ventanas.

Una vez explicado estos problemas de rendimiento de la aplicación Vic, le aconsejamos que visite la sección *Configuración recomendada para Vic* donde se le dará unos valores recomendados para obtener la mejor relación calidad/rendimiento en una sesión de Access Grid.

A.6.3. Configuración recomendada para Vic como Visualizador (Consumidor)

Aunque se han detallado cada una de las opciones que ofrece la aplicación Vic, en esta sección se pretende ofrecer una configuración recomendada que debería de funcionar en la mayoría de equipos de Sala. Dicha configuración se explicará de los tres métodos diferentes que existe para configurar la aplicación Vic. Después, se darán algunas configuraciones de ejemplo para tener una referencia de las posibles combinaciones y la potencia que ofrece el sistema Access Grid gracias al uso de los Servicios.

Dicha configuración se explicará de los tres métodos diferentes que existe para configurar la aplicación Vic.

Antes de empezar, asegúrese de que únicamente tiene instalado en su cliente de Access Grid el servicio de audio, AudioService, es decir, elimine todos los servicios de vídeo (VideoConsumer y derivados, VideoProducer y derivados) y deje únicamente el servicio de Audio.

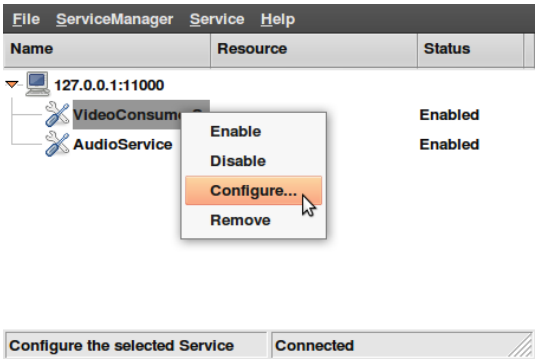
Si utiliza el Gestor de Servicios (Linux)

Si se va a configurar la aplicación Vic desde el propio Cliente de Sala de Access Grid, a través del **Gestor de Servicios del Nodo**, le recomendamos que configure el servicio de vídeo, tanto como Visualizador, de la siguiente manera:

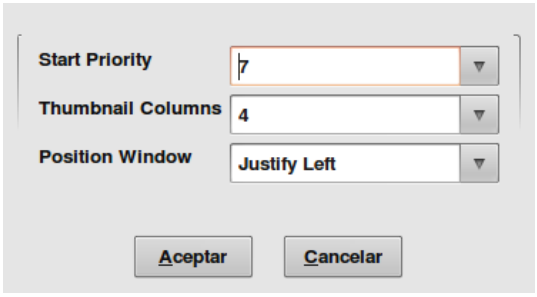
- Agregue, al cliente de Access Grid, el servicio **VideoConsumerH264** en la máquina Visualizadora. Dicho servicio capturará las fuentes de vídeo tanto si emiten utilizando el códec H.264/MPEG-4 como si emiten en H.261, por lo que, en una Sesión de Access Grid donde las fuentes de vídeo

sean de diversos códecs, con un único servicio podremos capturar la gran mayoría (a excepción de aquellos servicios extras como JPEGVideoProducerService o HDVideoProducerService que necesitan su *Consumer* correspondiente).

- Hacemos doble click en VideoConsumerService, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:



- Nos aparecerá la pantalla de configuración del visualizador de vídeo:

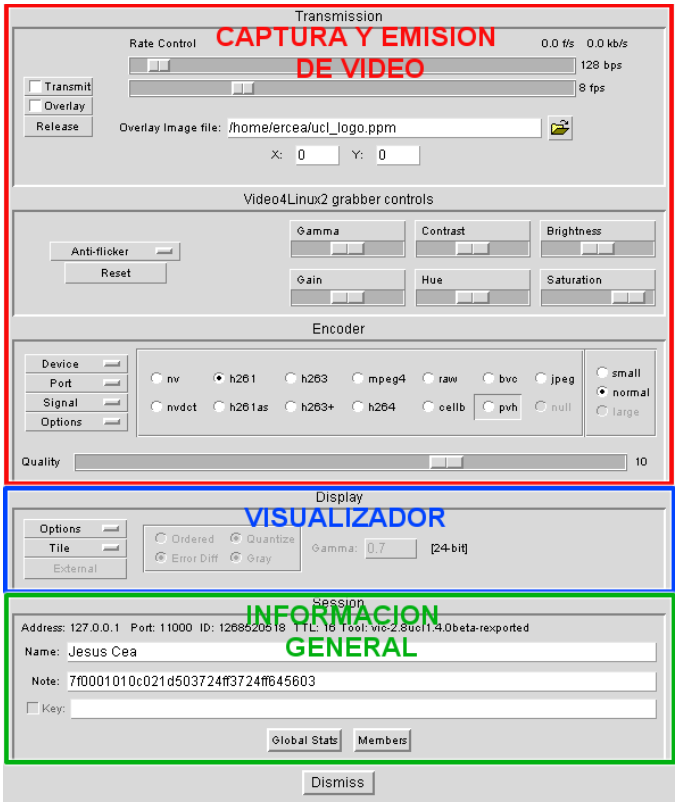


- Cambiamos el valor de **Thumbnail Columns** a **2** para distribuir las fuentes de vídeo en dos columnas.

- Pulsar botón **Aceptar**.

Si utiliza la propia aplicación Vic

Si se va a configurar la aplicación Vic recuerde que la aplicación está dividida en tres partes, nos centraremos en el recuadro **azul**:

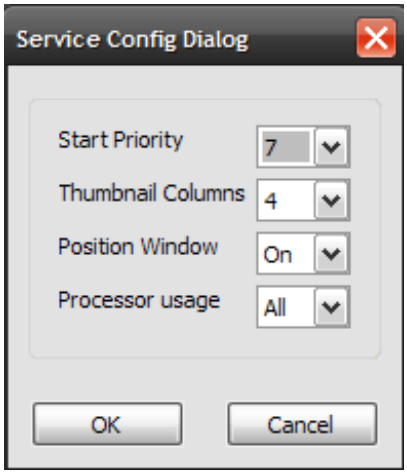


- **Option:** Marcar la opción Use Hardware Decode
- **Tile:** Marcar la opción Double

Si utiliza Windows

Si utiliza un sistema Windows como máquina Visualizadora es **muy recomendable** que configure una opción en el **Gestor de Servicios** de Access Grid para **solucionar un problema de rendimiento**:

- En las versiones Windows es importante y **muy recomendable** configurar un parámetro adicional que se incluye para optimizar el rendimiento:



- **Processor Usage:** Nos aseguraremos de que el valor establecido es 1.

- El resto de valores debe ser igual que en la versión Linux

Si utiliza un fichero de configuración

Si desea configurar la aplicación Vic, como **Consumidor**, a través de su fichero de configuración le recomendamos los siguientes valores:

```
[node]
servicemanagers = servicemanager0

[servicemanager0]
url =                #Si el consumidor esta en una maquina
                     #remota hay que indicarlo aqui con su IP
services = service_video_consumer #Mas otros servicios que tenga
                                  #en su configuracion
                                  #(service0 service1, etc.)
builtin = 1
name =

[service_video_consumer]
packageName = VideoConsumerService.zip
serviceConfig = service_video_consumer_config0

[service_video_consumer_config0]
Position Window = Justify Left
Start Priority = 7
Thumbnail Columns = 2
```

A.6.4. Configuración recomendada para Vic como Productor

Aunque se han detallado cada una de las opciones que ofrece la aplicación Vic, en esta sección se pretende ofrecer una configuración recomendada que debería de funcionar en la mayoría de equipos de Sala. Dicha configuración se explicará de los tres métodos diferentes que existe para configurar la aplicación Vic.

Se explicará, en su sección correspondiente, la configuración recomendada para el códec H.261 y, también, para el códec H.264/MPEG-4. Recuerde que estos cambios hay que realizarlos en aquella **máquina** cuya función sea la de **Emisión y Captura de Vídeo**.

Después, se darán algunas configuraciones de ejemplo para tener una referencia de las posibles combinaciones y la potencia que ofrece el sistema Access Grid gracias al uso de los Servicios. Dicha configuración se explicará de los tres métodos diferentes que existe para configurar la aplicación Vic.

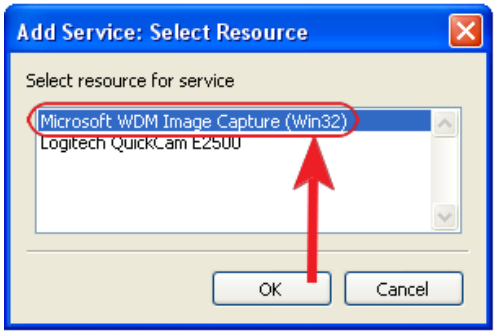
Antes de empezar, asegúrese de que únicamente tiene instalado en su cliente de Access Grid el servicio de audio, *AudioService*, es decir, elimine todos los servicios de vídeo (*VideoConsumer* y derivados, *VideoProducer* y derivados) y deje únicamente el servicio de Audio.

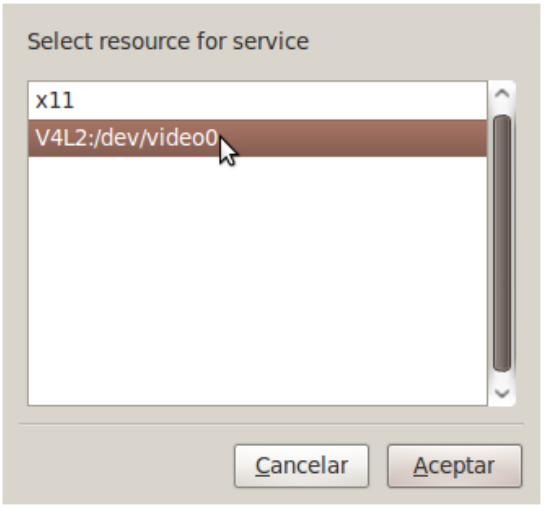
A.6.5. Configuración recomendada para Vic como Productor H.261

Si utiliza el Gestor de Servicios

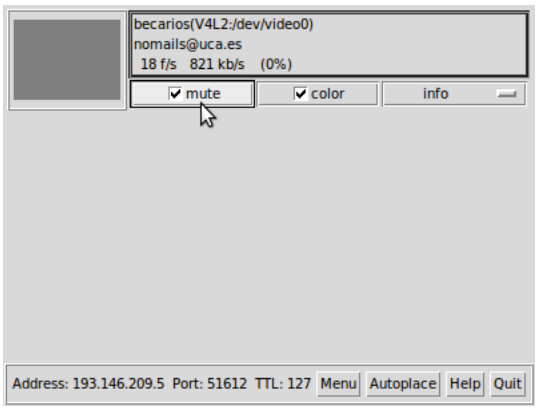
Si se va a configurar la aplicación Vic desde el propio Cliente de Sala de Access Grid, a través del **Gestor de Servicios del Nodo**, le recomendamos que configure el servicio de vídeo, como Captura y Emisión, de la siguiente manera:

- Este proceso hay que **repetirlo tantas veces como cámaras quiera agregar**.
    - Agregue, al cliente de Access Grid, el servicio **VideoProducerService** en la máquina de Emisión y Captura. Una vez agregado, le aparecerá una pantalla preguntándole qué fuente de vídeo desea capturar con dicho servicio: *Nota: En Linux puede que salga duplicado una misma cámara, una con **V4L**:, y otra con **V4L2**. En principio debería de seleccionar el **V4L2**, sin embargo, no todas las fuentes de vídeo son compatibles con este formato, por lo que deberá elegir **V4L**.*

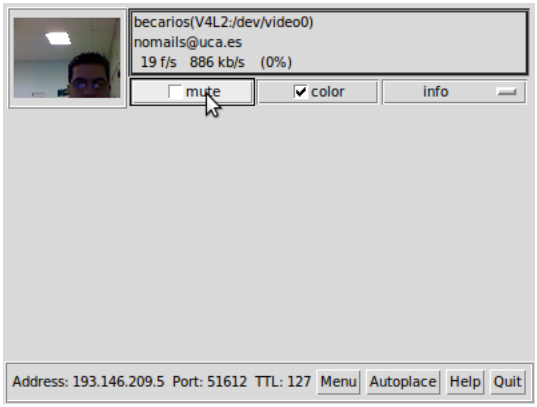




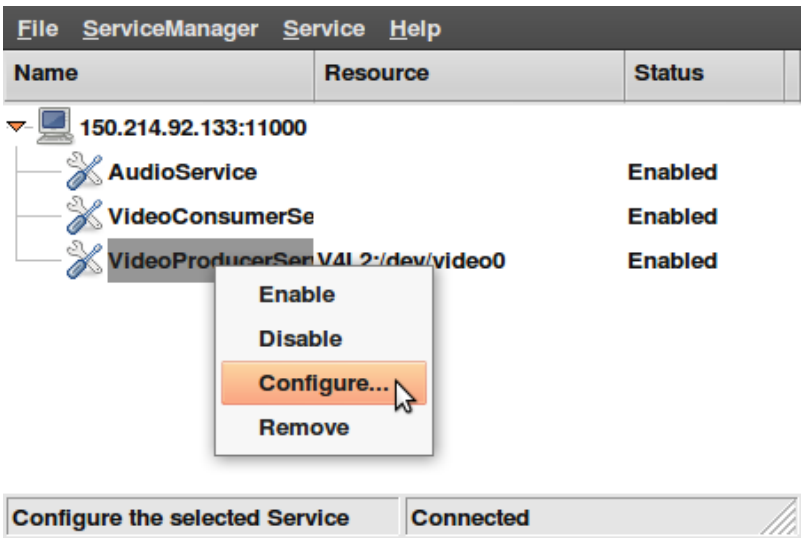
- Una vez acepte, se ejecutará la aplicación Vic con la fuente de vídeo añadida y muteada:



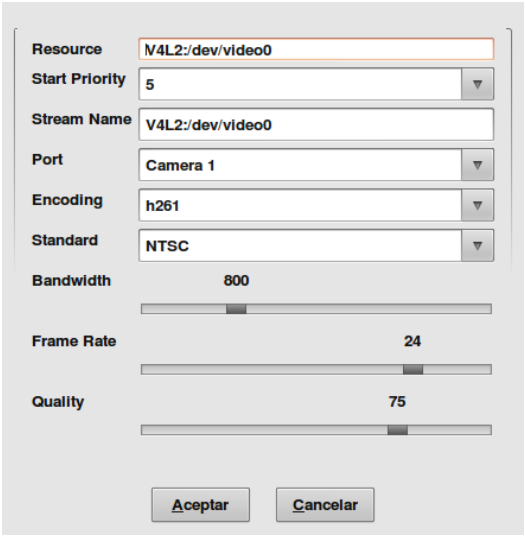
- Si aparece la ventana con la casilla Mute marcada, desmárquela para que empiece a emitir vídeo.



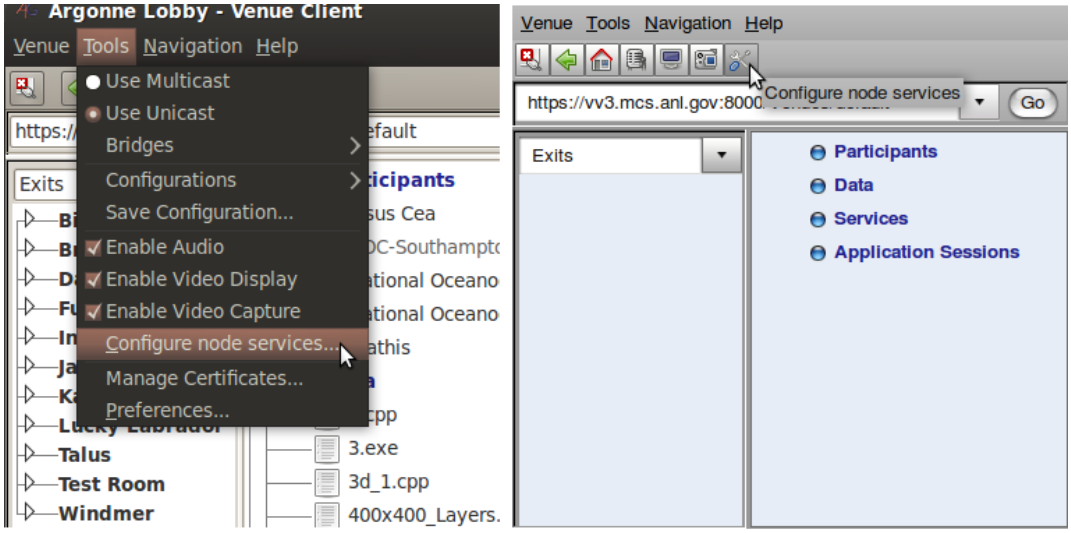
- Hacemos doble click en VideoProducerService, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:



- Nos aparecerá la pantalla de configuración del Vic, como Productor:

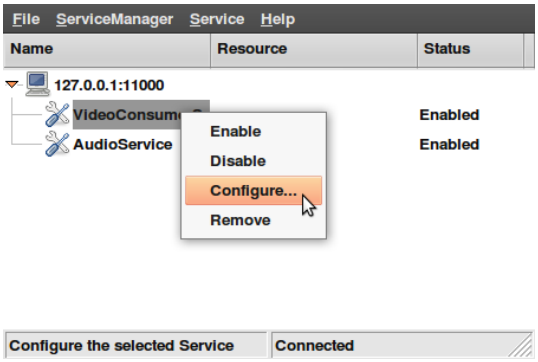


- Establecer los siguientes valores: *Nota: Recuerde que estos valores son los recomendados para el códec H.261, para el códec H.264/MPEG-4 se explicará a continuación.*
  - **Standard:** PAL
  - **Bandwith:** 800 kbps
  - **FrameRate:** 25 fps
  - **Quality:** 80 . A mayor valor, mayor calidad, pero el vídeo se ralentizará más. Consideramos que este valor es el adecuado
- Reiniciamos el servicio para que surtan efecto los cambios realizados. Recuerde reiniciar el servicio hay que hacer lo siguiente:
  - En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools − > Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



  - Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Seleccionamos el servicio de Visualizador de Vídeo

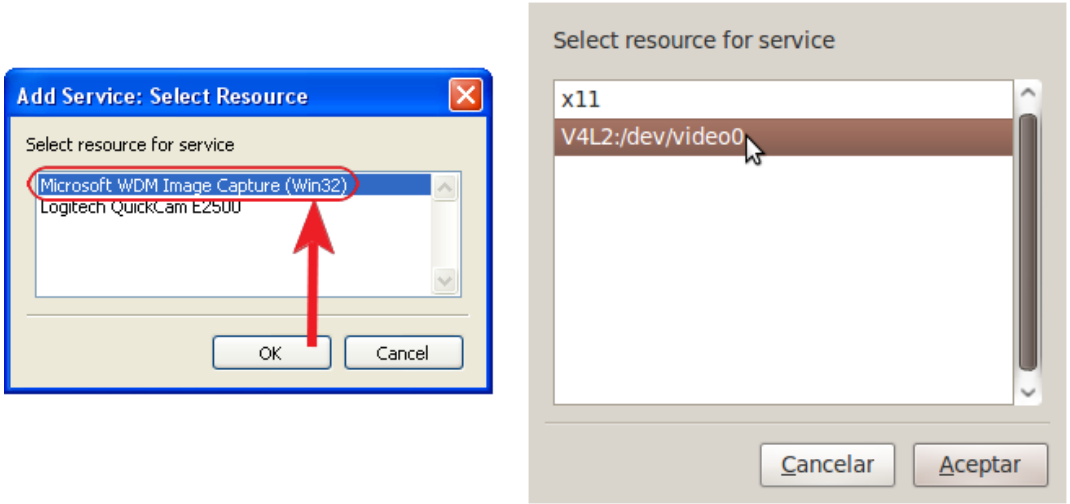
(***VideoProducerService***) y pulsamos el segundo botón del ratón. Le damos a ***Disabled*** para deshabilitar el servicio. Luego, repetimos el proceso para elegir, esta vez, ***Enabled***, habilitando así, de nuevo, el servicio de Visualizador de Vídeo:



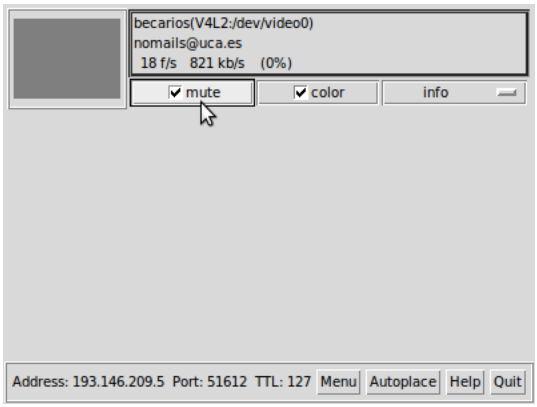
Si utiliza la propia aplicación Vic

- Este proceso hay que **repetirlo tantas veces como cámaras quiera agregar**.

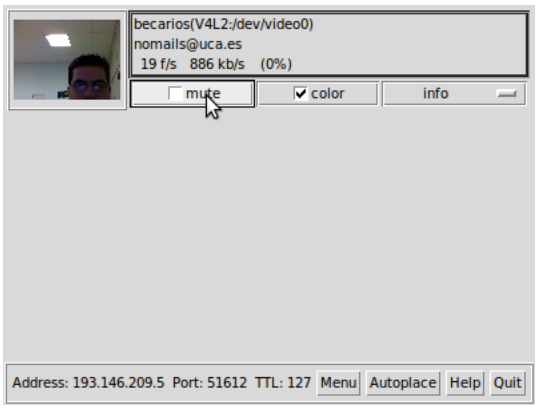
  - Agregue, al cliente de Access Grid, el servicio **VideoProducerService** en la máquina de Emisión y Captura. Una vez agregado, le aparecerá una pantalla preguntándole qué fuente de vídeo desea capturar con dicho servicio: *Nota: En Linux puede que salga duplicado una misma cámara, una con **V4L:**, y otra con **V4L2**. En principio debería de seleccionar el **V4L2**, sin embargo, no todas las fuentes de vídeo son compatibles con este formato, por lo que deberá elegir **V4L**.*



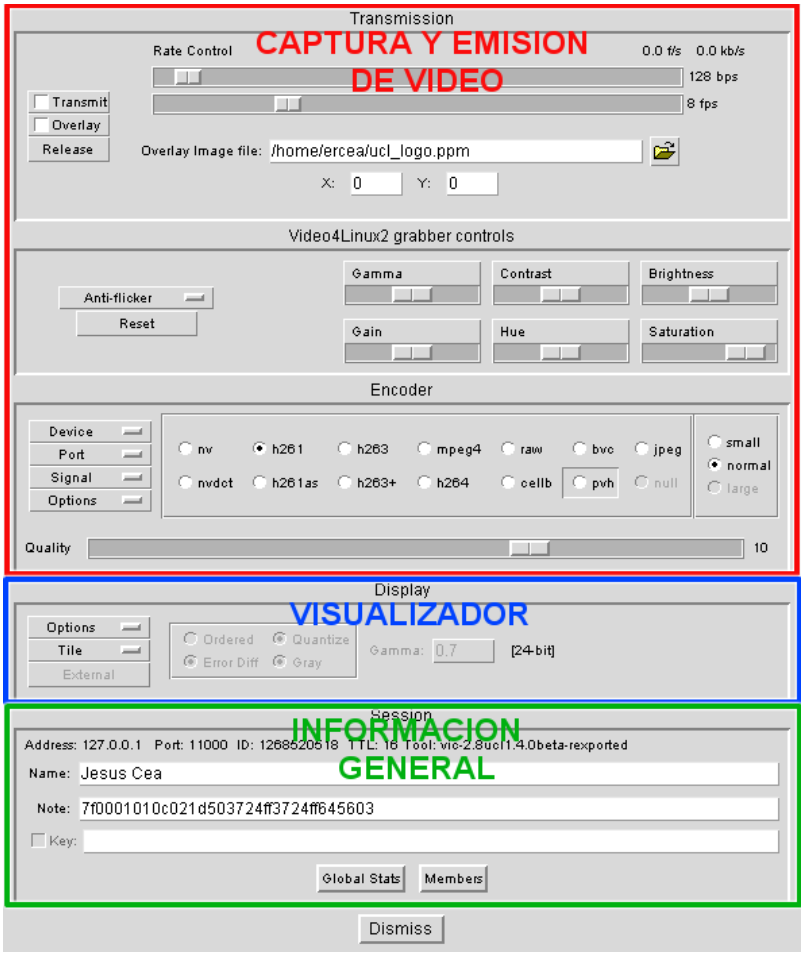
  - Una vez acepte, se ejecutará la aplicación Vic con la fuente de vídeo añadida y muteada:



- Si aparece la ventana con la casilla Mute marcada, desmárquela para que empiece a emitir vídeo.



- Diríjase a la aplicación Vic relacionada con la cámara añadida y pulse el botón Menu. Le aparecerá la pantalla de configuración. Recuerde que la aplicación está dividida tres partes, nos centraremos en el panel rojo:



- En **Rate Control** establezca el valor 800kbps y 25 fps
- Asegúrese que la opción **Transmit** está marcada.
- Asegúrese de que está marcado el códec **H.261** en la lista de códecs.
- Deslice la barra **Quality** hasta que marque el valor 6.
- En la opción **Signal** elija PAL.
- En Options marque **Use Hardware Encode**.

Si utiliza un fichero de configuración

Si desea configurar la aplicación Vic, como **Productor H.261**, a través de su fichero de configuración le recomendamos los siguientes valores:

```
[node]
servicemanagers = servicemanager0

[servicemanager0]
url =  #Si la fuente de video esta en una maquina remota
       #hay que indicarlo aqui con su IP
services = service_video_producerH261_0 service_video_producerH261_1 ...
           #Mas otros servicios que tenga en su configuracion
builtin = 1
name =
```

```
11
12  [service_video_producerH261_0]
13  packageName = VideoProducerService.zip
14  resource = resource_H261_0
15  serviceConfig = service_video_producerH261_config0
16
17  [resource_H261_0]
18  name = V4L2:/dev/video0
19         #Este es un ejemplo, pero debe de indicar
20         #la ruta de su fuente de video
21
22  [service_video_producerH261_config0]
23  Frame Rate = 25
24  Encoding = h261
25  Start Priority = 5
26  Standard = PAL
27  Bandwidth = 800
28  Port = zc3xx
29  Quality = 80
30  Stream Name = V4L2:/dev/video0
31                #Debe ser el mismo que se
32                #haya indicado en [resource_H261_0]
33
34  # Anadimos otra camara
35
36  [service_video_producerH261_1]
37  packageName = VideoProducerService.zip
38  resource = resource_H261_1
39  serviceConfig = service_video_producerH261_config1
40
41  [resource_H261]
42  name = V4L2:/dev/video1
43         #Este es un ejemplo, pero debe de
44         #indicar la ruta de su fuente de video
45
46  [service_video_producerH261_config1]
47  Frame Rate = 25
48  Encoding = h261
49  Start Priority = 5
50  Standard = PAL
51  Bandwidth = 800
52  Port = zc3xx
53  Quality = 80
54  Stream Name = V4L2:/dev/video1
55                #Debe ser el mismo que se
56                #haya indicado en [resource_H261_1]
```

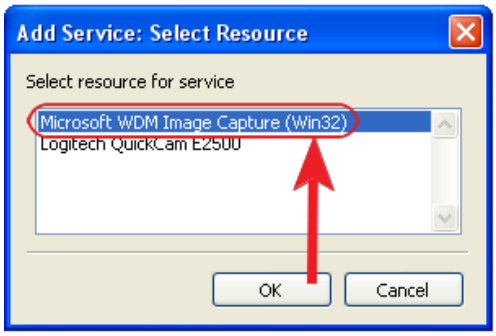
A.6.6. Configuración recomendada para Vic como Productor MPEG-4/H.264

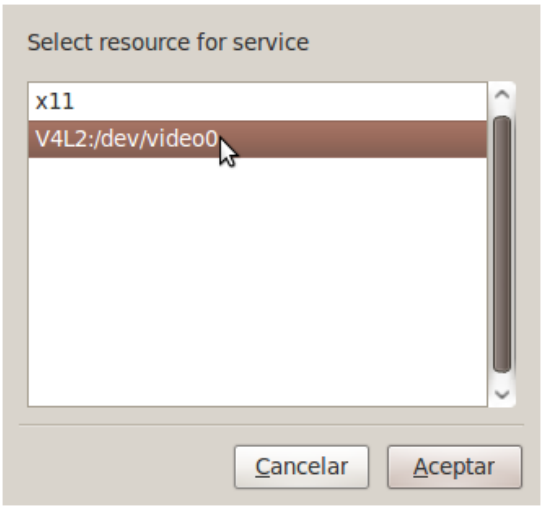
Si utiliza el Gestor de Servicios

Si se va a configurar la aplicación Vic desde el propio Cliente de Sala de Access Grid, a través del **Gestor de Servicios del Nodo**, le recomendamos que configure el servicio de vídeo, como Captura y Emisión, de la siguiente manera:

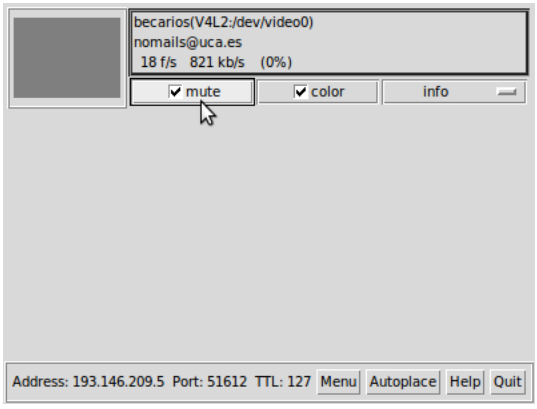
- Este proceso hay que **repetirlo tantas veces como cámaras quiera agregar**.

  - Agregue, al cliente de Access Grid, el servicio **VideoProducerServiceH264** en la máquina de Emisión y Captura. Una vez agregado, le aparecerá una pantalla preguntándole qué fuente de vídeo desea capturar con dicho servicio: *Nota: En Linux puede que salga duplicado una misma cámara, una con **V4L:**, y otra con **V4L2**. En principio debería de seleccionar el **V4L2**, sin embargo, no todas las fuentes de vídeo son compatibles con este formato, por lo que deberá elegir **V4L**.*

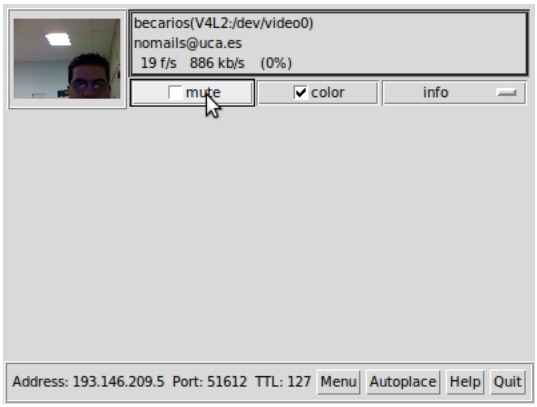




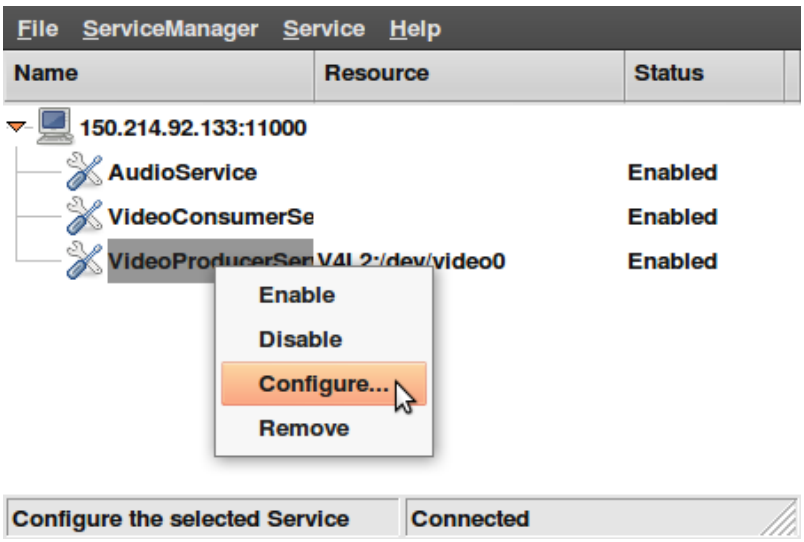
  - Una vez acepte, se ejecutará la aplicación Vic con la fuente de vídeo añadida y muteada:



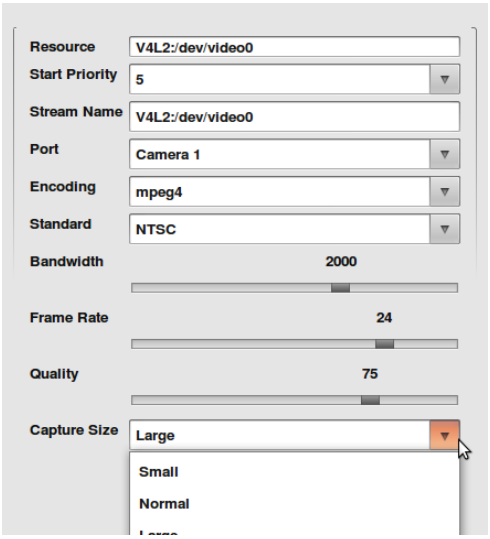
  - Si aparece la ventana con la casilla Mute marcada, desmárquela para que empiece a emitir vídeo.



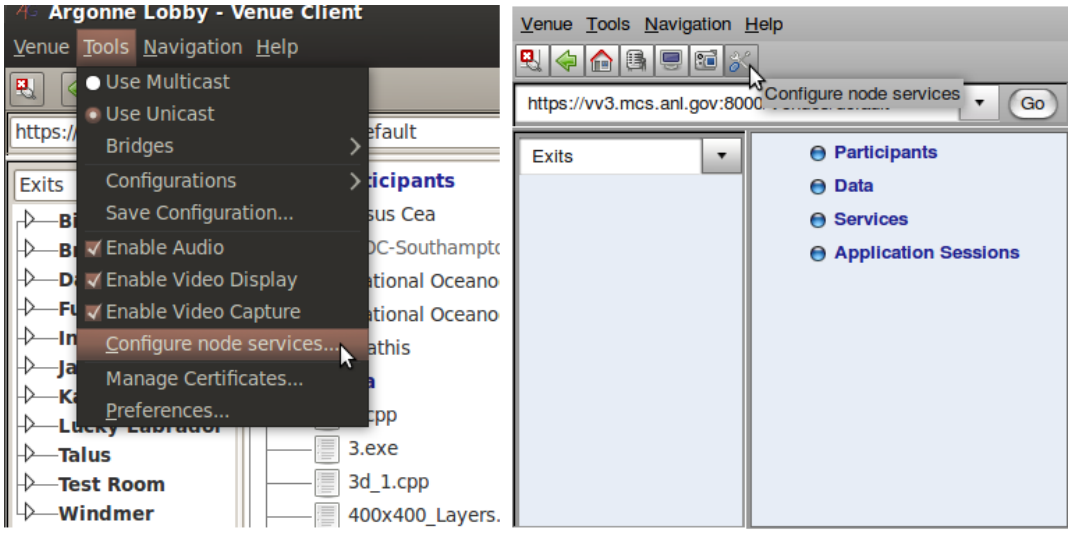
- Hacemos doble click en *VideoProducerServiceH264*, o bien, lo seleccionamos, pulsamos el segundo botón del ratón y elegimos **Configure...**:



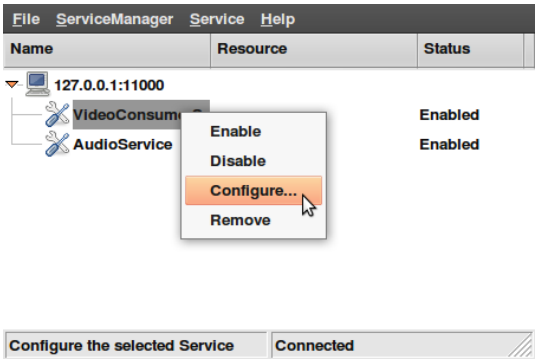
- Nos aparecerá la pantalla de configuración del Vic, como Productor:



- Establecer los siguientes valores: *Nota: Recuerde que estos valores son los recomendados para el códec H.261, para el códec H.264/MPEG-4 se explicará a continuación.*
  - **Encoding:** MPEG-4 (H.264 consume más recursos y la diferencia de calidad no es notoria)
  - **Standard:** PAL
  - **Bandwith:** 800 kbps
  - **FrameRate:** 25 fps
  - **Quality:** 80 . A mayor valor, mayor calidad, pero el vídeo se ralentizará más. Consideramos que este valor es el adecuado
  - **Capture Size:** Medium
- Reiniciamos el servicio para que surtan efecto los cambios realizados. Recuerde reiniciar el servicio hay que hacer lo siguiente:
  - En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



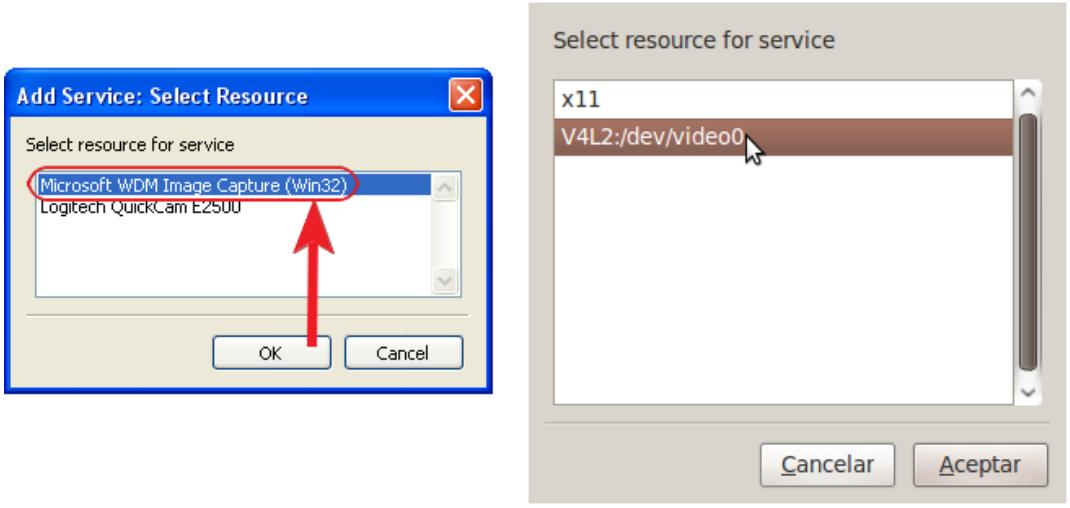
  - Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Seleccionamos el servicio de Visualizador de Vídeo (***VideoProducerServiceH264***) y pulsamos el segundo botón del ratón. Le damos a ***Disabled*** para deshabilitar el servicio. Luego, repetimos el proceso para elegir, esta vez, ***Enabled***, habilitando así, de nuevo, el servicio de Visualizador de Vídeo:



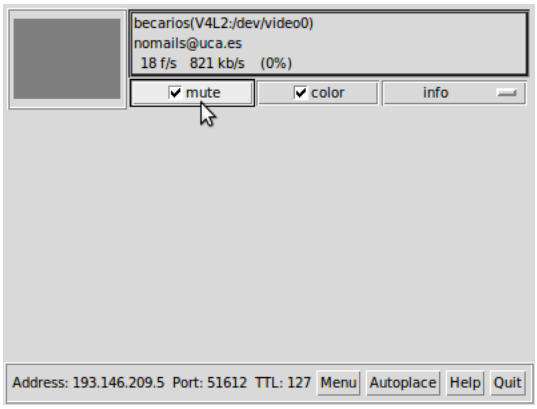
Si utiliza la propia aplicación Vic

- Este proceso hay que **repetirlo tantas veces como cámaras quiera agregar**.

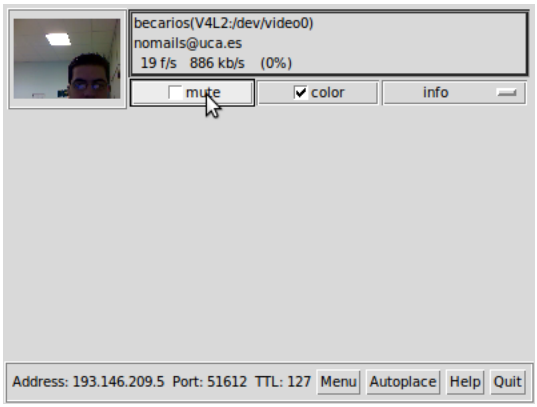
  - Agregue, al cliente de Access Grid, el servicio **VideoProducerServiceH264** en la máquina de Emisión y Captura. Una vez agregado, le aparecerá una pantalla preguntándole qué fuente de vídeo desea capturar con dicho servicio: *Nota: En Linux puede que salga duplicado una misma cámara, una con **V4L**:, y otra con **V4L2**. En principio debería de seleccionar el **V4L2**, sin embargo, no todas las fuentes de vídeo son compatibles con este formato, por lo que deberá elegir **V4L**.*



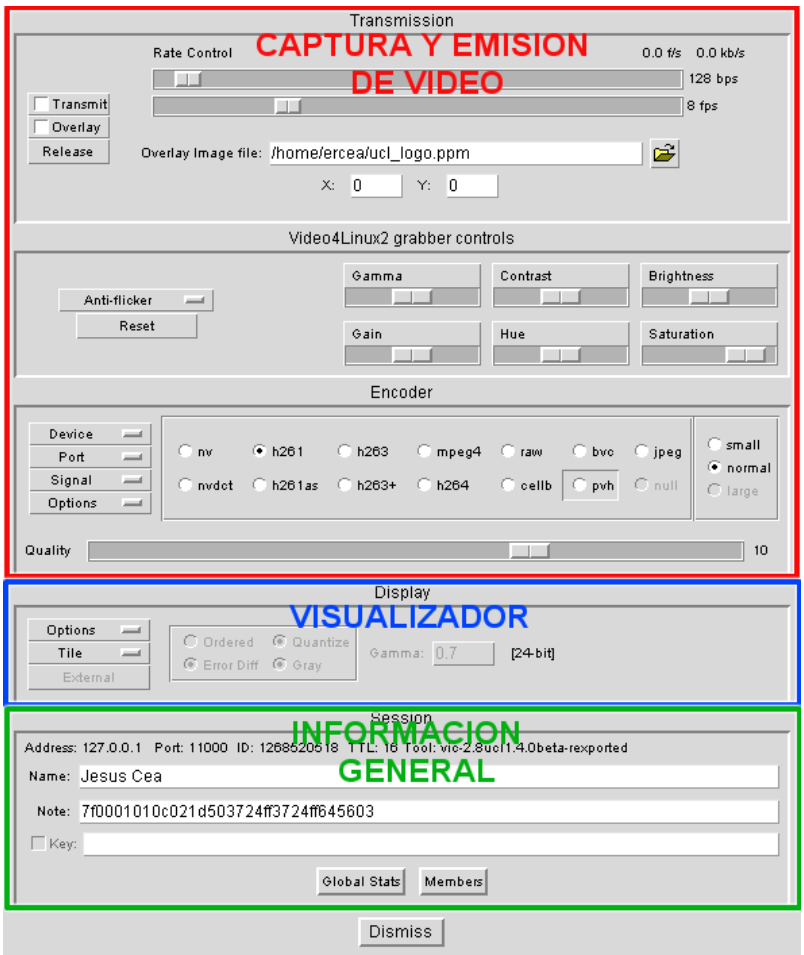
  - Una vez acepte, se ejecutará la aplicación Vic con la fuente de vídeo añadida y muteada:



  - Si aparece la ventana con la casilla Mute marcada, desmárquela para que empiece a emitir vídeo.



- Diríjase a la aplicación Vic relacionada con la cámara añadida y pulse el botón Menu. Le aparecerá la pantalla de configuración. Recuerde que la aplicación está dividida tres partes, nos centraremos en el panel rojo:



  - En **Rate Control** establezca el valor 800kbps y 25 fps
  - Asegúrese que la opción **Transmit** está marcada.
  - Asegúrese de que está marcado el códec **MPEG-4** en la lista de códecs.
  - En MPEG-4/H.264 la barra **Quality está deshabilitada**.
  - En la opción **Signal** elija PAL.

- En Options marque **Use Hardware Encode**.

Si utiliza un fichero de configuración

Si desea configurar la aplicación Vic, como **Productor MPEG-4/H.264**, a través de su fichero de configuración le recomendamos los siguientes valores:

```
[node]
servicemanagers = servicemanager0

[servicemanager0]
url =   #Si la fuente de video esta en una maquina
        #remota hay que indicarlo aqui con su IP

services = service_video_producerMpeg4_0, service_video_producerMpeg4_1 ...
          #Mas otros servicios que tenga en su configuracion

builtin = 1
name =

[service_video_producerMpeg4_0]
packageName = VideoProducerServiceH264.zip
resource = resource_mpeg4_0
serviceConfig = service_video_producerMpeg4_config0

[resource_mpeg4_0]
name = V4L2:/dev/video0
       #Este es un ejemplo, pero debe de indicar
       #la ruta de su fuente de video

[service_video_producerMpeg4_config0]
Frame Rate = 25
Encoding = mpeg4
Start Priority = 5
Standard = PAL
Capture Size = Normal
Bandwidth = 800
Port = zc3xx
Quality = 80
Stream Name = V4L2:/dev/video0
              #Debe ser el mismo que se haya
              #indicado en [resource_mpeg4_0]

# Anadimos otra camara

[service_video_producerMpeg4_1]
packageName = VideoProducerServiceH264.zip
resource = resource_mpeg4_1
serviceConfig = service_video_producerMpeg4_config1

[resource_mpeg4_1]
name = V4L2:/dev/video1
       #Este es un ejemplo, pero debe de
       #indicar la ruta de su fuente de video

```

```
[service_video_producerMpeg4_config1]
Frame Rate = 25
Encoding = mpeg4
Start Priority = 5
Standard = PAL
Capture Size = Normal
Bandwidth = 800
Port = zc3xx
Quality = 80
Stream Name = V4L2:/dev/video1
                #Debe ser el mismo que se
                #haya indicado en [resource_mpeg4_1]
```

A.7. Configuración de Audio (*RAT*)

Es cierto que en una videoconferencia es muy importante el que exista una buena calidad de vídeo puesto que, al tratarse de una videoconferencia, los participantes pueden verse entre ellos y, así, se podría simular como si una reunión presencial real se tratara. Sin embargo, el sonido se puede considerar mucho más importante que el vídeo, ya que, el objetivo de toda videoconferencia es la de transmitir información, y la forma más habitual de transmitirla es mediante el uso de conversaciones. Por lo tanto, no tener sonido o tener una mala configuración de él, por muy bien que llegue a verse los participantes entre ellos, si no pueden comunicarse correctamente o ni si quiera se pueden comunicar, no tendría sentido la reunión.

En este apartado se intentará dar unas simples nociones relacionadas con el sonido, con el objetivo de que pueda conseguir el mejor sonido posible en una reunión.

Se explicará brevemente el uso y configuración de la aplicación RAT (*Robust Audio Tool*), la aplicación dedicada a la emisión y recepción de audio. Además, se hablará de un problema bastante común en el ámbito del sonido, el eco, donde se explicará el motivo de que se produzca y algunas soluciones para reducirlo o eliminarlo.

Por último, en la última sección, se dará unas breves indicaciones o nociones sobre qué pasos hay que seguir para una correcta configuración de audio en una Sala de VideoConferencia.

A.7.1. Configuración de RAT

La aplicación RAT (Robust Audio Tool) viene integrada en el sistema Access Grid. Cuando entramos en una sesión de Access Grid, si tenemos añadido, o queremos añadir, un servicio de audio, ***AudioService***, se ejecutará automáticamente para gestionar el audio, tanto la emisión, como la recepción. Podemos configurar RAT de tres formas:

- A través del Gestor de Servicios, "Tools $->$ Configure node services", que será la recomendada, ya que, los cambios que hagamos desde aquí, podrán ser guardados en un fichero de configuración para no tener que repetir el procedimiento. Además, desde aquí podemos configurar los servicios que estén alojados en otra máquina (a través de ServiceManager, ver *Añadir Servicios situados en otra máquina*), y, como antes, también podremos guardar la configuración en un fichero de configuración.
- Desde el propio *RAT*: Se configura directamente desde la propia aplicación. Recomendado para configuraciones concretas, ya que los cambios que se haga no podrán ser guardados en un fichero

de configuración. Sin embargo, al tratarse de la propia aplicación, permite una configuración más avanzada que la que ofrece el Gestor de Servicios.

- Crear/Editar un fichero de configuración: Access Grid permite guardar ficheros de configuración, donde se almacenan las configuraciones de los servicios que usemos (audio, vídeo, etc.) en nuestra sesión de Access Grid. Si uno de los servicios que usamos es el *AudioService*, podemos editar nuestro fichero para configurar aquellas opciones que ofrece RAT pero no están disponibles desde el Gestor de Servicios. En capítulos posteriores se explicará con detalle cómo editar o crear un fichero de configuración personalizado.

Configuración a través del Gestor de Servicios

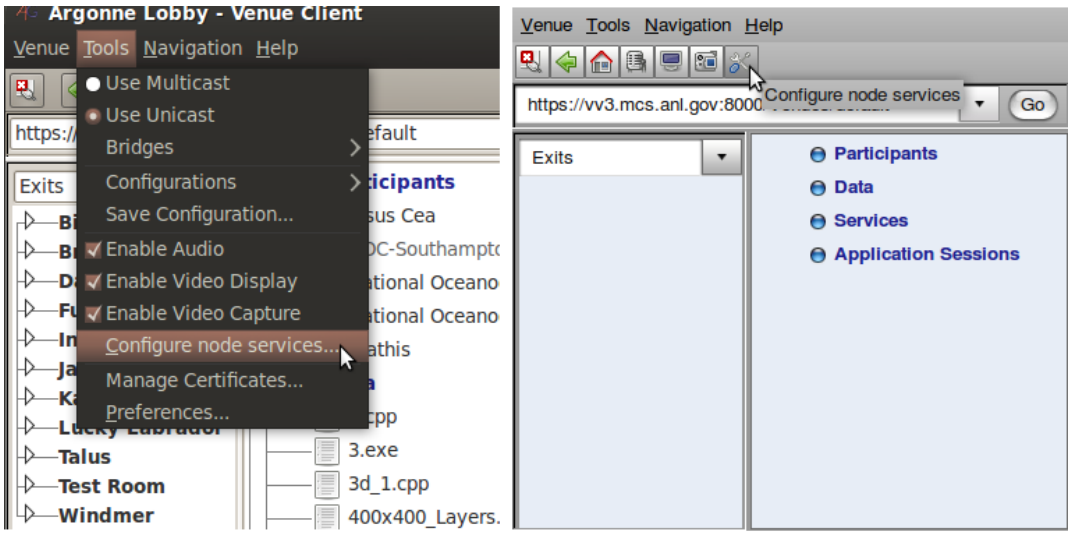
Uno de los métodos para poder configurar RAT es a través del **Gestor de Servicios del Nodo** de Access Grid. RAT es una aplicación independiente que ha sido integrada en el sistema Access Grid, así que, desde este método, nos aparecerá una interfaz sencilla con las opciones que se han considerado relevantes para el buen funcionamiento del Audio en Access Grid.

Por otro lado, si el servicio de audio se encuentra añadido en otra máquina distinta a la de visualización (como es nuestro caso), desde la máquina de visualización podremos configurar dicho servicio remotamente a través de este método, sin necesidad de tener que acceder a la otra máquina. Es decir, el Gestor de Servicios nos permite configurar servicios alojados en otras máquinas de forma remota.

Sin embargo, utilizar este método de configuración tiene el inconveniente de que los cambios que hagamos en la configuración no surtirán efecto a no ser que, o bien deshabilitemos y habilitemos el servicio de Audio (*AudioService*), o bien, volviendo a entrar en la Sala.

Para configurar *RAT* a través del Gestor de Servicios, hacemos lo siguiente:

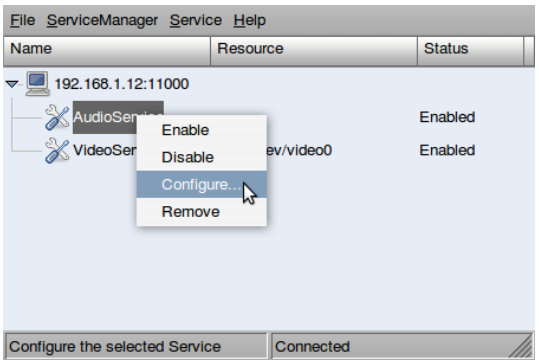
- En el Cliente de Sala (VenueClient), nos dirigimos a (Tools ->Configure node services), también podemos pulsar el último icono de la interfaz del Cliente de Sala:



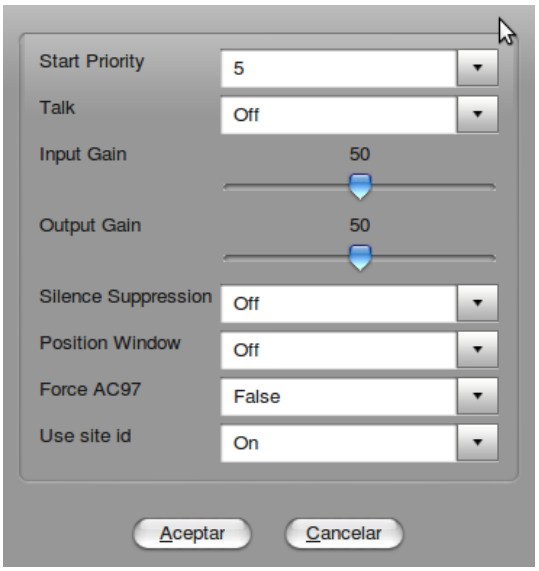
- Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Hacemos doble click en ***AudioService***, o bien, lo seleccionamos,

pulsamos el segundo botón del ratón y elegimos **Configure...**:

Nota: Hay que recordar que hay que tener añadido el servicio VideoConsumerService, en caso de que no aparezca en el listado, hay que añadirlo.



- Nos aparecerá la pantalla de configuración de audio:



- Las opciones que podemos configurar son:

  - **Start Priority** [**Valores:** *1 al 10*, **Por defecto:** *5*]: Esta opción nos permite variar la prioridad de este proceso, a mayor prioridad, más uso de CPU se le asigna.
  - **Talk** [**Valores:** *Off / On*, **Por defecto:** *Off*]: Esta opción permite activar o desactivar la emisión de sonido. Si dejamos el valor en Off, nadie podrá escuchar lo que hablemos por nuestro micrófono, en cambio, si ponemos el valor On, activaremos nuestro micrófono y los participantes de la sala podrán oírnos.
  - **Input Gain** [**Valores:** *0-100*, **Por defecto:** *50*]: Esta opción cambia el volumen del sonido que recibimos, es decir, del sonido de los participantes de la sala. A mayor valor, mayor volumen y, por tanto, se oirá más fuerte.
  - **Output Gain** [**Valores:** *0-100*, **Por defecto:** *50*]: Esta opción cambia el volumen del sonido que emitimos, es decir, el sonido de nuestro micrófono. A mayor valor, mayor volumen

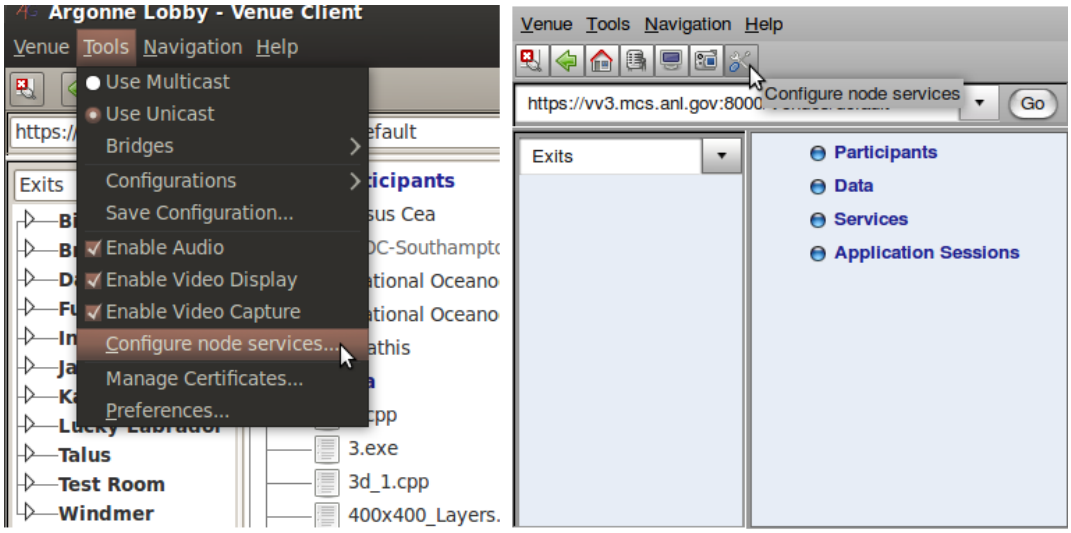
y, por tanto, más fuerte nos oirán los participantes de la Sala. Hay que tener cuidado si establecemos valores muy altos, ya que podría distorsionar nuestra voz, o incluso emitir interferencias producidas por nuestro micrófono al resto de participantes.

- **Silence Supression** [**Valores:** *Off / Automatic / Manual*, **Por defecto:** *Off*]: Esta opción habilita o deshabilita un efecto para eliminar el silencio a través de software. Si elegimos el valor On activaremos dicho efecto.
- **Position Window** [**Valores:** *Off / Justify Left / Justify Right*, **Por defecto:** *Off*]: Esta opción posiciona automáticamente la ventana de la aplicación *RAT*. Si elegimos el valor *Off* la posición será la que nosotros hayamos puesto. Si elegimos el valor *Justify Left*, colocará la ventana a la izquierda, y si elegimos el valor *Justify Right* colocará la ventana a la derecha.
- **Force AC97** [**Valores:** *False / True*, **Por defecto:** *False*]: Esta opción activa un parche para solventar problemas con las tarjetas de sonido AC97. Es recomendable activarlo en el caso de que haya problemas de sonido, si disponemos una tarjeta de esta gama.
- **Use Site Id** [**Valores:** *Off / On*, **Por defecto:** *On*]: Esta opción habilita o deshabilita el envío de *SiteId*, que es una cadena de texto arbitraria para proporcionarnos una identificación en la red de forma automática.

Cómo hacer que los cambios realizados surtan efecto

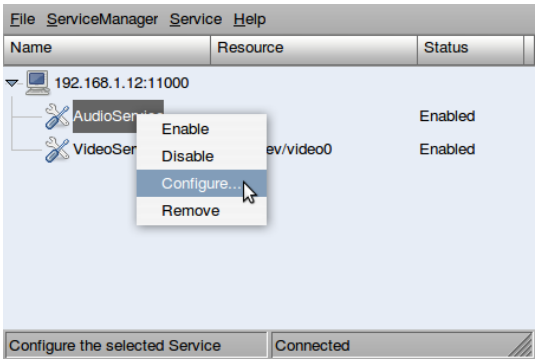
Como se ha comentado anteriormente, el inconveniente que tiene utilizar este método es que, una vez realizado los cambios oportunos, no surtirán efecto a no ser que recarguemos el servicio de Visualizador de Vídeo, o volvamos a entrar en la sala.

- **Volver a entrar a la sala:** Simplemente, volviendo a pulsar sobre el botón Go volveremos a entrar en la sala.
- **Recargar el servicio de Audio:**
  - En el Cliente de Sala (*VenueClient*), nos dirigimos a "Tools $->$ Configure node services", también podemos pulsar el último icono de la interfaz del Cliente de Sala:



- Nos aparecerá la ventana del Gestor de Servicios con el listado de servicios que tenemos añadido en nuestra configuración. Seleccionamos el servicio de Visualizador de Vídeo (***AudioService***)

y pulsamos el segundo botón del ratón. Le damos a ***Disabled*** para deshabilitar el servicio. Luego, repetimos el proceso para elegir, esta vez, ***Enabled***, habilitando así, de nuevo, el servicio de Visualizador de Vídeo:



Hay que recordar que, una vez realizado los cambios oportunos, según nuestras necesidades, podemos guardarlas en un fichero de configuración en Access Grid. Esto se explicará en un próximo capítulo, ya que, dicho fichero de configuración no guarda únicamente la configuración del vídeo, sino una configuración global de nuestros servicios añadidos en Access Grid.

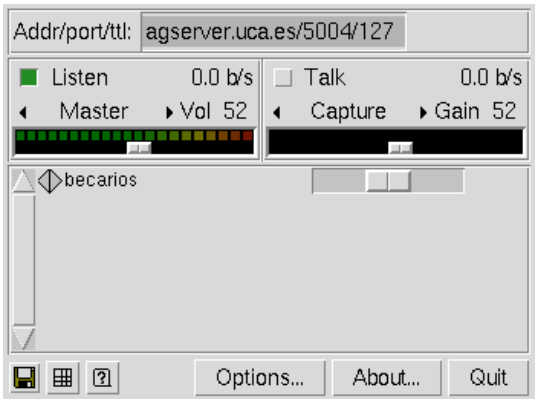
Configuración desde RAT

Otro de los posibles métodos de configurar la aplicación *RAT* es configurarla directamente desde la propia aplicación. Este método es directo ya que se está configurando desde la propia aplicación y, además, a diferencia del método anterior, los cambios que se hagan toman efecto en el mismo instante. Además ofrece muchas más opciones de configuración, por lo que da más versatilidad a la configuración del audio.

Sin embargo, utilizar este método tiene algunos inconvenientes. Por un lado, los cambios que hagamos en él no podremos guardarlo para futuras sesiones, por lo que tendremos que repetir el procedimiento cada vez que entremos en una sala (a no ser que guardemos o creemos un fichero de configuración, en próximos capítulos se explicará cómo). Además, al tratarse de la propia aplicación, ofrece numerosas opciones a configurar, pero muchas de ellas no son de utilidad para lo que vamos a necesitar de él en Access Grid, por lo que, para el usuario principiante o normal, puede parecerle más difícil de configurar.

No obstante, para los usuarios avanzados que quieran experimentar y/o exprimir al máximo la configuración de audio, se detallarán todas las opciones que ofrece esta aplicación.

Como ya se sabe, una vez iniciada una sesión de Access Grid y, si tenemos el servicio de audio añadido a nuestra lista de servicios (dentro del *Gestor de Servicios del Nodo*), nos debe aparecer, entre otras cosas, la aplicación *RAT*:



Cabe destacar que esta aplicación aparecerá en aquella máquina donde esté el servicio añadido. Es decir, si, por ejemplo, tenemos dos máquinas, una para la visualización y otra para la captura de vídeo y emisión/transmisión de audio (que es nuestro caso), la aplicación *RAT* deberá aparecer en esta última máquina. Por lo que, si queremos configurar el audio a través de *RAT*, tendremos que acceder a la máquina que tenga añadido este servicio y veremos la ventana de *RAT* como en la imagen anterior.

En resumen, este servicio **debe ser añadido en la máquina que se vaya a encargar de la gestión del audio**.

Desde esta pantalla tenemos a nuestra disposición las siguientes opciones:

- **Listen:** Este botón activa/desactiva la posibilidad de escuchar al resto de participantes de la Sala. Si está en gris, la opción estará deshabilitada y no podremos oír a nadie. En cambio, si está en verde, la opción está habilitada y podremos oír a los participantes de la Sala. Con la barra de desplazamiento inferior a dicho botón podremos establecer el volumen al que queremos oír a los participantes de la Sala (es el equivalente a la opción *Input Gain* del Gestor de Servicios del método anterior). El valor de esta barra se muestra al lado del texto **Vol**.

- $<$ **Master** $>$: Pulsando sobre las flechas que están a los lados de la palabra **Master** cambiaremos la fuente de audio que queremos oír. Una vez cambiada, podemos cambiar el volumen con la misma barra de desplazamiento inferior a éste.

- **Tasa de trasferencia de recepción (Kb/s):** Muestra la cantidad de información de audio, en Kb/s, que estamos recibiendo a través de la red.

- **Talk:** Esta opción permite activar o desactivar la emisión de sonido. Si está en gris, nadie podrá escuchar lo que hablemos por nuestro micrófono, en cambio, si está en verde, activaremos nuestro micrófono y los participantes de la sala podrán oírnos. Con la barra de desplazamiento inferior a dicho botón podremos establecer el volumen al que queremos oír a los participantes de la Sala (es el equivalente a la opción *Output Gain* del Gestor de Servicios del método anterior). El valor de esta barra se muestra al lado del texto **Gain**. Hay que tener cuidado si establecemos valores muy altos, ya que podría distorsionar nuestra voz, o incluso emitir interferencias producidas por nuestro micrófono al resto de participantes.

- $<$ **Capture** $>$: Pulsando sobre las flechas que están a los lados de la palabra **Capture** cambiaremos la fuente de audio que se desea capturar para enviar a la Sala (si tenemos más de una). Una vez cambiada, podemos cambiar el volumen con la misma barra de desplazamiento inferior a éste.

- **Tasa de transferencia de envío (Kb/s):** Muestra la cantidad de información de audio, en Kb/s, que estamos enviando a la red.

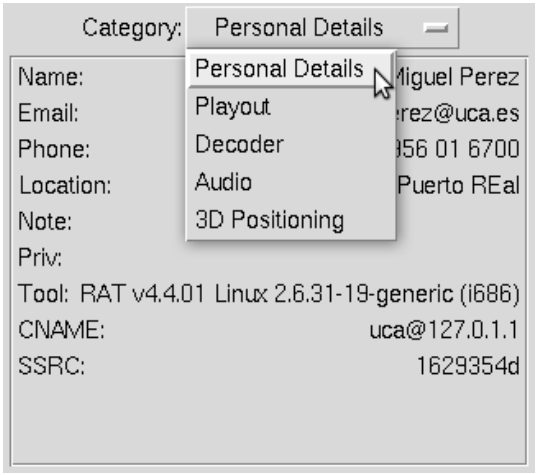
- **Lista de participantes:** En la parte central de la ventana aparecen los participantes que están actualmente conectados a la Sala (en el caso particular de RAT serían los usuarios conectados a una dirección IP concreta). El formato de esta lista es:

  - **Iconos Listen/Talk (por confirmar):** Corresponde al rombo, dividido por la mitad, que está a la izquierda del nombre del participante. Según el color (gris o verde) indica si el usuario tiene la opción Listen activada/desactivada (para la primera mitad), y si tiene la opción Talk activada/desactivada (para la segunda mitad).

  - **Nombre del participante:** Si pulsamos sobre él, nos aparecerá información sobre su perfil. Si pulsamos el botón central del ratón, silenciaremos a dicho participante (no podremos oírle).

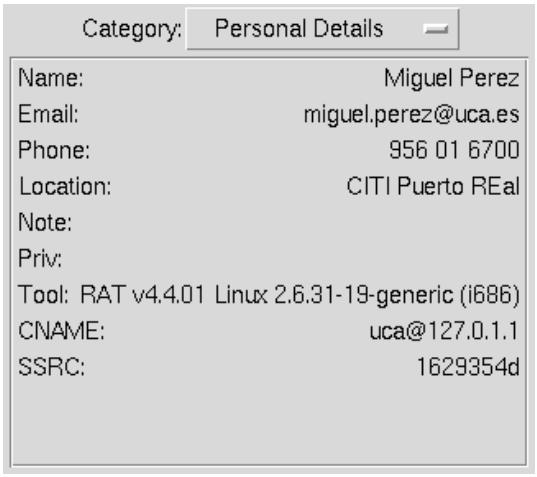
  - **Barra de volumen individual:** Esta barra permite establecer un volumen individual para cada participante. A más a la derecha, más volumen tendrá y, por tanto, más fuerte se oirá a ese participante. Es útil en sesiones donde existan participantes que se escuchen muy bajo y, así, poder aumentar su volumen.

  - **Haciendo click en el nombre:** Aparecerá un panel donde se muestra varias opciones para ese usuario concreto:

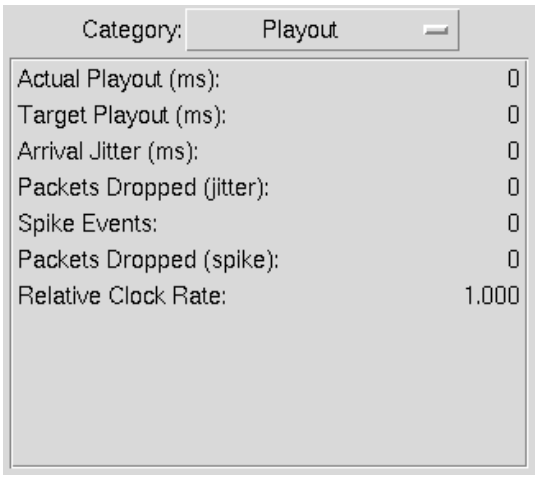


Podemos observar que está dividido en varias secciones:

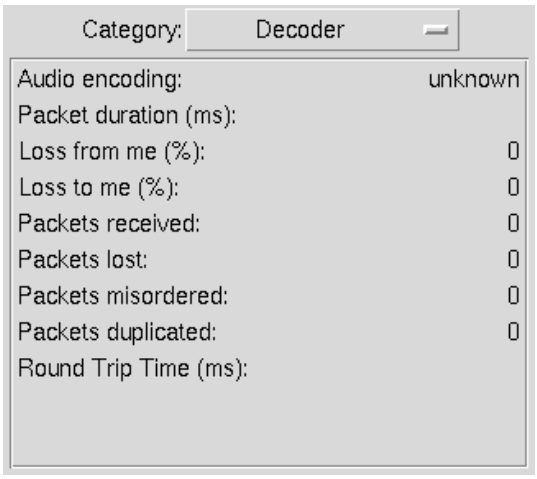
- **Personal Details:** Muestra información sobre el perfil del usuario.



- **Playout:** Muestra información sobre el audio del usuario seleccionado.

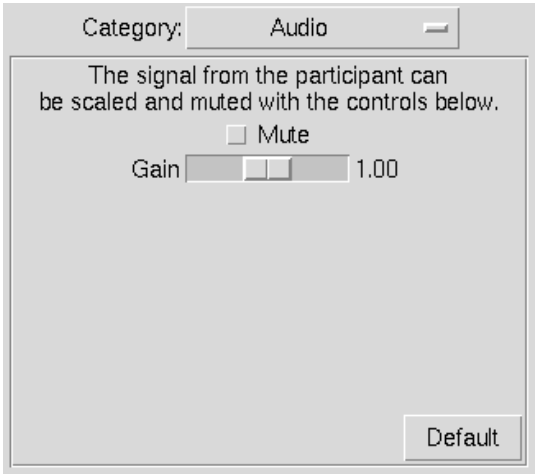


- **Decoder:** Muestra información sobre el códec utilizado además de información relacionada con la calidad (paquetes perdidos, porcentaje de pérdida, etc.)

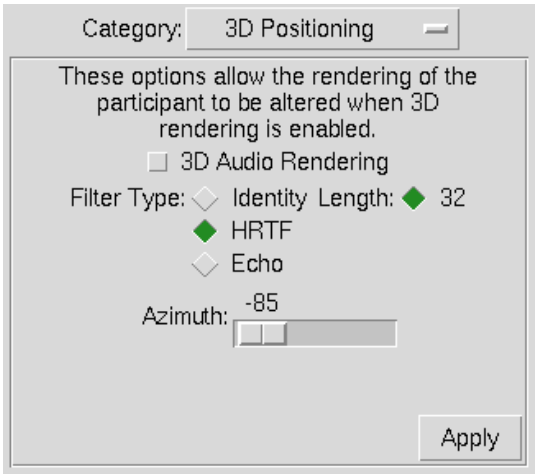


- **Audio:** En esta sección podemos silenciar al usuario seleccionado pulsando el botón Mute. También podemos aumentar o disminuir su volumen moviendo la barra de volu-

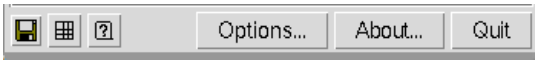
men Gain.



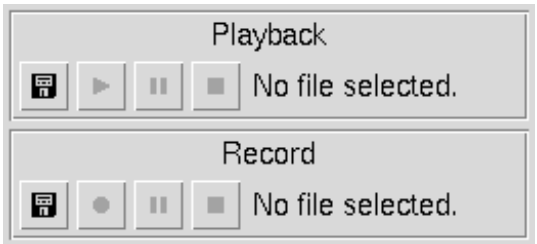
- **3D Positioning:** Muestra varias opciones para el posicionado de sonido 3D.



- Barra de Botones:

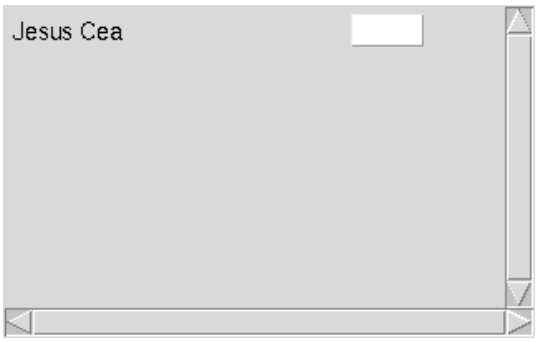


- **Reproducir/Guardar Sesión (icono guardar):** Nos aparecerá una ventana donde podremos grabar la sesión actual en un fichero de audio, o reproducir una sesión anteriormente grabada.



- **Matriz de ganancia:** Muestra una ventana donde aparece un listado con todos los participantes actualmente conectados a la sesión y la calidad de recepción de todos ellos, indicán-

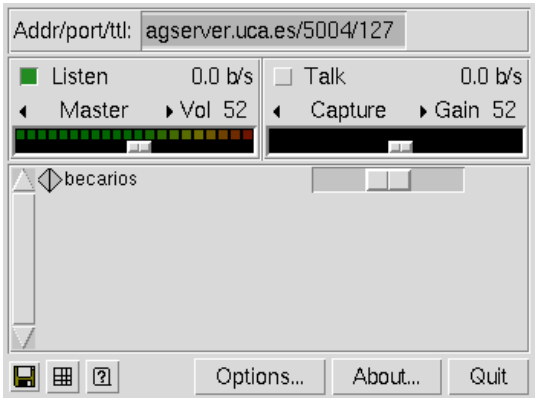
dolo a través de un porcentaje que aparece a la derecha de cada participante. A mayor valor del porcentaje, mejor es la calidad de recepción.



- **Activar/Desactivar Ayuda:** Si está activada (botón pulsado), aparecerán cuadros de textos emergentes (*balloon helps*) donde se proporciona información de ayuda en relación a cualquier objeto de RAT que se esté señalando con el ratón.
- **Botón Options:** Muestra la ventana de configuración avanzada de RAT. Este menú se explicará en la sección Configuración Avanzada de RAT.
- **Botón About:** Muestra información acerca de la aplicación RAT (versión, página oficial, etc.).
- **Botón Quit:** Cierra la aplicación RAT

A.7.2. Configuración Avanzada de RAT

En este apartado se detallará cada una de las opciones que ofrece la aplicación RAT que disponemos al pulsar el botón Options de la interfaz principal del programa:



Hay que aclarar que, **para una sesión correcta de Access Grid, en principio, no es necesario entrar ni configurar nada de este apartado**, ya que, las opciones descritas en el apartado anterior, son suficientes para poder establecer una correcta configuración de audio en una sesión de Access Grid. Además, por defecto, este apartado viene configurado para ofrecer la mayor calidad de audio posible (salvo alguna excepción).

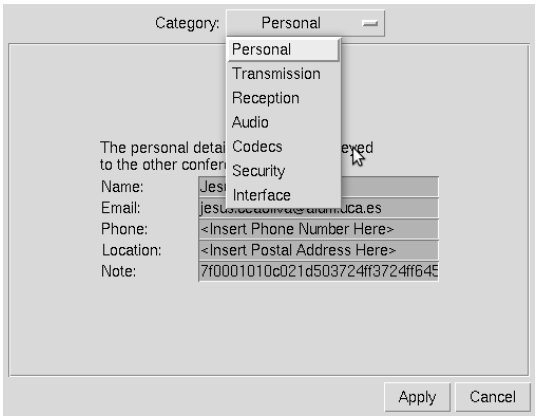
Esta sección está destinada a usuarios experimentados que quieran probar otras opciones para aprovechar al máximo las capacidades de esta aplicación y obtener el mejor audio posible en su sala, o en el caso

de que tengan problemas de audio o de rendimiento en su PC.

Una vez aclarado, pasaremos a detallar cada una de las opciones que ofrece la aplicación RAT.

Categorías

Cuando se pulsa el botón **Options** en la interfaz principal de la aplicación RAT, nos aparecerá por defecto la siguiente pantalla:



Como puede observar, dicha ventana está dividida en dos partes, donde una parte engloba a la otra.

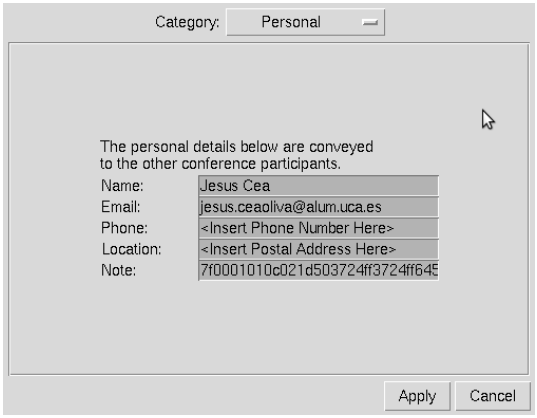
- **Category:** La primera parte corresponde a la selección de la categoría de opciones que desea configurar. Por defecto, aparece la categoría Personal, sin embargo, pulsando sobre dicha categoría, aparecerá un menú desplegable con todas las categorías que ofrece la aplicación RAT a configurar. Después, en la parte inferior aparecen dos botones:
  - **Apply:** Para aplicar los cambios que se realicen.
  - **Cancel:** Para cancelar los cambios y volver a la ventana principal de la aplicación RAT.
- **Contenido de la categoría:** Es un panel donde se recopila el conjunto de opciones relacionadas con la categoría seleccionada.

Las categorías que tenemos disponibles son (*depende de la versión de la aplicación RAT que disponga*):

- Personal
- Transmission
- Reception
- Audio
- Codecs
- Security
- Interface

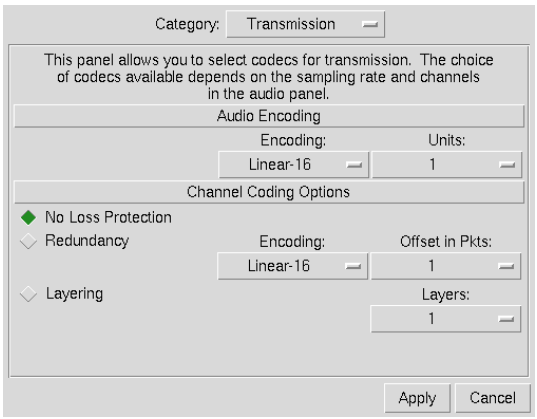
A continuación se detalla cada una de las categorías y sus opciones que ofrece la aplicación RAT.

Personal



Esta categoría nos permite configurar nuestra información de perfil de usuario. Al estar integrado en Access Grid, dicha información se autocompleta con el perfil introducido la primera vez que ejecuta el Cliente de Sala de Access Grid. No obstante, usted puede cambiar sus datos desde este panel (también puede cambiarlo en el Cliente de Sala y éste, automáticamente, lo rellena en este panel).

Transmission



Esta categoría nos ofrece un conjunto de opciones para configurar la transmisión de nuestro audio. Podemos diferenciar de este panel dos partes:

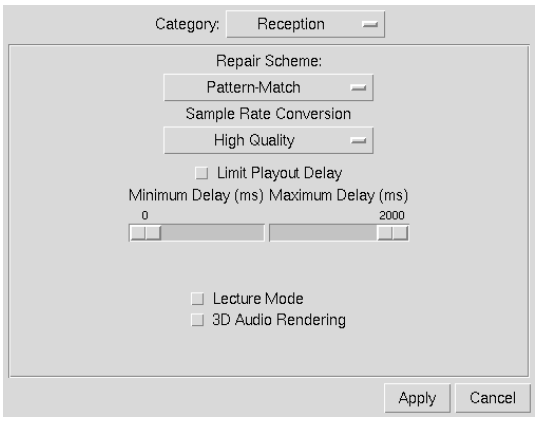
- **Audio Encoding:** Conjunto de opciones para configurar la codificación del audio. Las opciones son:
  - **Encoding** [**Valores:** *Linear16 / ?-law / A-law / G726-40 / G726-32 / G726-24 / G726-16 / DVI / VDVI / WBS / GSM*, **Por defecto:** *Linear16*]: Cambia el esquema primario de compresión de audio. Esta lista está ordenada con el códec que utiliza la mejor calidad y más consumo de ancho de banda como primero de la lista, a peor calidad y menos consumo de ancho de banda como último de la lista.
  - **Units** [**Valores:** *De 1 a 62 en base 2*, **Por defecto:** *1*]: Establece la duración de cada paquete de audio enviado. Existe un consumo constante de paquetes, así que, elevando este valor se

reduciría el consumo total. Sin embargo, el efecto de paquetes perdidos es más notable con paquetes grandes.

- **Channel Coding Options:** Conjunto de opciones para configurar la codificación del canal de audio. Las opciones son:

  - **Filtros de canal:** Existen los siguientes filtros:
    - **No Loss Protection:** No establece ninguna codificación en el canal.
    - **Redundancy:** Transporta unidades de audio recientes en paquetes para evitar la pérdida de paquetes. Algunas herramientas de audio (como por ejemplo, vat-4.0) no están preparadas para este tipo de paquetes, teniendo esta opción marcada.
    - **Layering:** Por determinar...
  - **Encoding** [**Valores:** *Linear16 / ?-law / A-law / G726-40 / G726-32 / G726-24 / G726-16 / DVI / VDVI / WBS / GSM*, **Por defecto:** *Linear16*]: Establece el formato de los datos transportados. Esta lista está ordenada con el códec que utiliza la mejor calidad y más consumo de ancho de banda como primero de la lista, a peor calidad y menos consumo de ancho de banda como último de la lista.
  - **Offset in Pkts** [**Valores:** *1 al 8 en base 2*, **Por defecto:** *1*]: Establece el offset de los datos transportados.
  - **Layers (Sólo para códec WBS)** [**Valores:** *1*, **Por defecto:** *1*]: Establece el número de capas que serán enviadas. Es necesario ejecutar RAT desde la línea de comandos o desde una terminal con la opción "-l n <dir>/<puerto> <dir>/<puerto> ..." , donde n es el número de capas y hay que escribir tantos <dir>/<puerto> como capas se hayan establecido.

Reception



Esta categoría nos ofrece un conjunto de opciones para configurar la recepción del audio. Podemos encontrar las siguientes opciones:

- **Repair Scheme** [**Valores:** *Pattern-Match / Repeat / Noise / None*, **Por defecto:** *Pattern-Match*]: Establece el tipo de corrección que se aplicará cuando haya una pérdida de paquetes. El listado aparece en orden incremental de complejidad y calidad de la corrección (el primero es el más complejo y de mejor calidad de corrección).

- **Sample Rate Conversion** [**Valores:** *High Quality / Intermediate Quality / Low Quality*, **Por defecto:** *High Quality*]: Establece el tipo de tasa de frecuencia de muestreo que se aplicará a aquellos flujos de audio que difieran su tasa de frecuencia de muestreo a la tasa de frecuencia de muestreo de la tarjeta de sonido.

- **Limit Playout Delay:** Activando esta opción, obliga a RAT a que establezca los límites de retardo que se establecen con las barras que aparecen debajo de esta opción. Normalmente no es deseable activar esta opción.

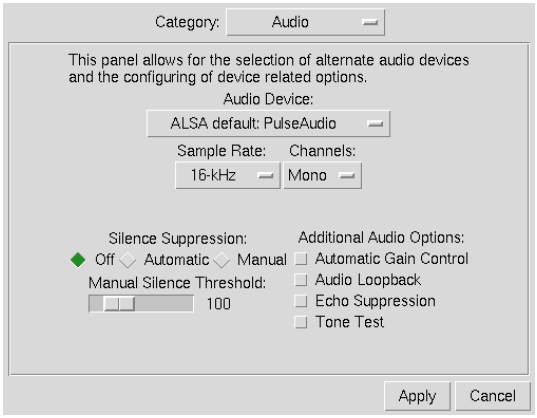
- **Minimum Delay (ms)** [**Valores:** *0-1000*, **Por defecto:** *0*]: Establece el retardo mínimo que será aplicado en los flujos de audio de entrada. A más a la derecha, mayor valor y, por tanto, mayor límite de retardo mínimo. Este valor tomará efecto si se activa la opción **Limit Playout Delay**.

- **Maximum Delay (ms)** [**Valores:** *1000-2000*, **Por defecto:** *2000*]: Establece el retardo máximo que será aplicado en los flujos de audio de entrada. A más a la derecha, mayor valor y, por tanto, mayor límite de retardo mínimo. Este valor tomará efecto si se activa la opción **Limit Playout Delay**.

- **Lecture Mode:** Si se activa esta opción se añadirá retardo extra tanto en el envío como en la recepción de audio. Como su nombre indica, esta opción es recomendable para aquellos escenarios donde se transmite en una sola dirección (lectura, conferencia, etc.) y el resto escucha, donde la interactividad es menos importante que la calidad.

- **3D Audio Rendering:** Si se activa, habilita el renderizado de audio 3D, simulando dicho efecto de sonido 3D.

Audio



Esta categoría nos ofrece un conjunto de opciones para seleccionar y configurar la tarjeta de sonido. Podemos encontrar las siguientes opciones:

- **Audio Device** [**Valores:** *ALSA default: Pulse Audio / ALSA 1-HDA Intel / OSS:Realtek ALC268 / No Audio Device*, **Por defecto:** *ALSA default: Pulse Audio*]: Selecciona la tarjeta de sonido. Si se elige *No Audio Device* no seleccionará ninguna tarjeta de sonido.

- **Sample Rate** [**Valores:** *8, 11, 16, 22, 32, 44, 48 kHz* , **Por defecto:** *16 kHz*]: Establece la frecuencia de muestreo de la tarjeta de sonido. Esto afecta a los códecs elegidos en los paneles anteriores.

- **Channels** [**Valores:** *Mono / Stereo*, **Por defecto:** *Mono*]: Cambia el tipo de canal entre mono (un sólo canal) y estéreo (dos canales).

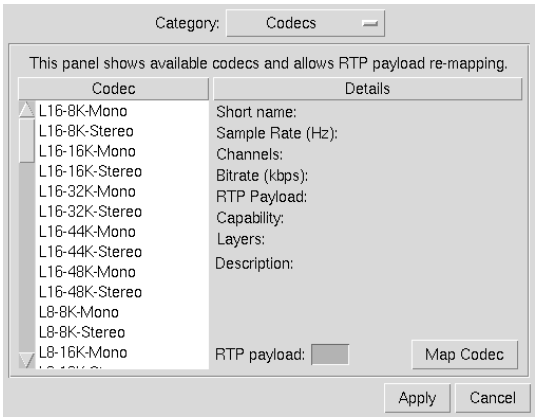
- **Silence Supression** [**Valores:** *Off / Automatic / Manual*, **Por defecto:** *Off*]: Establece el tipo de supresión de silencio que se aplicará.

  - **Off:** Todo audio es transmitido cuando la entrada no está silenciada (no muteada).
  - **Automatic:** Sólo el audio que sobrepase un umbral automáticamente establecido es transmitido cuando la entrada no está silenciada (no muteada).
  - **Manual:** Sólo el audio que sobrepase un umbral establecido manualmente es transmitido cuando la entrada no está silenciada (no muteada). Este umbral se establece con la siguiente opción.
  - **Manual Silence Threshold (*válido si se activa Manual*)** [**Valores:** *1-500* , **Por defecto:** *100*]: Establece el umbral a sobrepasar.

- **Additional Audio Options:** Conjunto de opciones adicionales para la tarjeta de sonido. Estas opciones son:

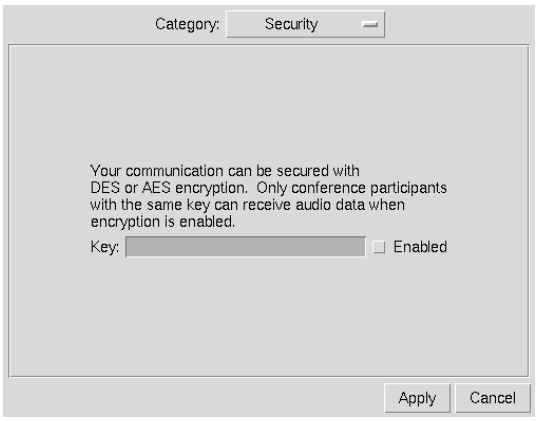
  - **Automatic Gain Control:** Si se activa, automáticamente ajusta el control de volumen del audio que se envíe.
  - **Audio Loopback:** Habilita el hardware para retroalimentar la entrada de audio.
  - **Echo Supression:** Si se activa, silencia el micrófono cuando se reproduce audio.
  - **Tone Test:** Emite un tono de sonido para testear el buen funcionamiento de la tarjeta de sonido. Este sonido no se transmite.

Codecs



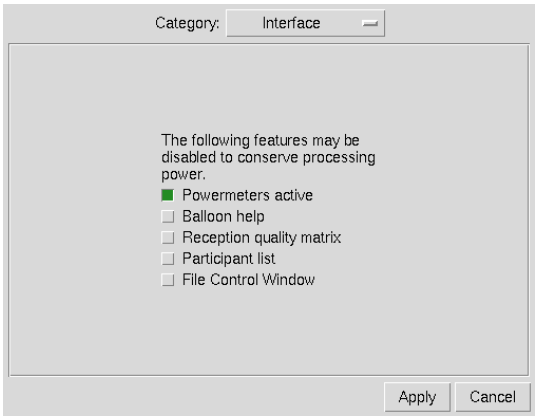
Esta categoría nos ofrece un listado de códecs disponibles para la aplicación RAT. Se puede observar que la ventana está dividida en dos partes. En la parte izquierda aparecen todos los códecs disponibles. Cuando selecciona un códec, automáticamente se rellena toda la información, referente al códec seleccionado, en la parte derecha (nombre del códec, frecuencia de muestreo, canales, etc.). Únicamente podemos cambiar el valor **RTP Playload**, seleccionando el códec al que se le quiera modificar dicho valor y luego pulsando el botón **Map Codec**. Sin embargo, este valor no es recomendable cambiarlo y dejarlo como viene por defecto.

Security



En esta categoría encontramos una opción cuya funcionalidad es encriptar todo el flujo de audio utilizando uno de los algoritmos de encriptación, AES o DES. Únicamente aquellos participantes que tengan la misma clave establecida recibirán el audio cuando esta opción esté activa. Para activarla, basta con escribir una clave (ha de ser la misma en todos los participantes) en el cuadro de texto de **Key** y, por último, pulsar el botón **Enable**.

Interface



Esta categoría nos ofrece un conjunto de opciones que nos permite configurar la interfaz de la aplicación *RAT*. Según la propia aplicación, recomienda desactivar estas opciones para no sobrecargar el procesador y obtener un mejor rendimiento. Podemos encontrar las siguientes opciones:

- **Powermeters active:** Esta opción activa (en verde) o desactiva (en gris) las barras de potencia de los volúmenes tanto de recepción como de emisión (las que están debajo de las opciones **Listen** y **Talk**). Esta opción debería desmarcarse si posee un PC con poca potencia.

- **Balloon help:** Si está activada (botón pulsado en verde), aparecerán cuadros de textos emergentes (*balloon helps*) donde se proporciona información de ayuda en relación a cualquier objeto de RAT que se esté señalando con el ratón.

- **Reception quality matrix:** Muestra (en verde) u oculta (en gris) la ventana de *Matriz de ganancia* explicada en la sección anterior.
- **Participant list:** Muestra (en verde) u oculta (en gris) la lista de participantes conectados en la interfaz principal de RAT.
- **File Control Window:** Muestra (en verde) u oculta (en gris) la ventana de Reproducir/Guardar Sesión explicada en la sección anterior.

A.7.3. Aplicar los cambios

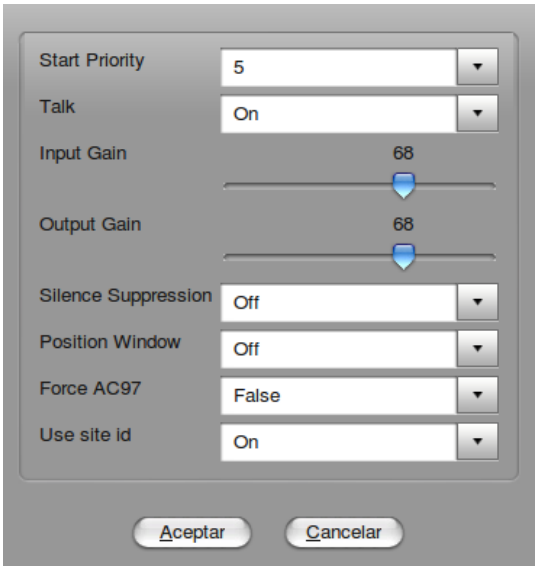
Todo cambio que se realice en estos paneles no surtirá efecto hasta que se pulse el botón **Apply**. En cambio, si se pulsa el botón **Cancel** se cancelarán todos los cambios realizados desde la última vez que fueron aplicados.

A.7.4. Configuración Recomendada para RAT

Aunque se han detallado cada una de las opciones que ofrece la aplicación RAT, en esta sección se pretende ofrecer una configuración recomendada que debería de funcionar en la mayoría de equipos de Sala. Dicha configuración se explicará de los tres métodos diferentes que existe para configurar la aplicación RAT.

Si utiliza el Gestor de Servicios del Nodo

Si se va a configurar la aplicación RAT desde el propio Cliente de Sala de Access Grid, a través del **Gestor de Servicios del Nodo**, le recomendamos que configure el servicio de audio (*AudioService*) de la siguiente manera:

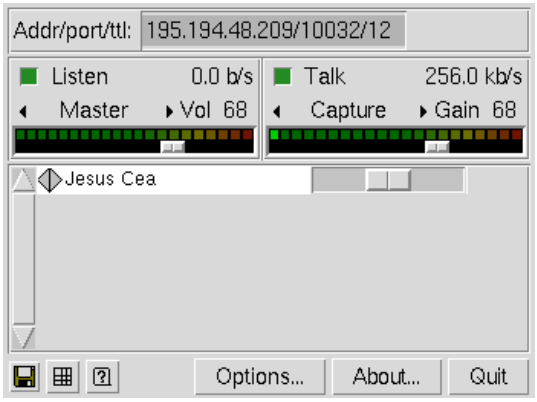


- **Talk:** On
- **Input Gain:** Entre 60 y 70 (68, por ejemplo)
- **Output Gain:** Entre 60 y 70 (68, por ejemplo)
- **Silence Supression:** Off

- **Position Window:** Off
- **Force AC97:** Off (a no ser que tenga una tarjeta de sonido de esta gama y esté teniendo problemas)
- **Use site id:** On

Si utiliza la propia aplicación RAT

Si se va a configurar la aplicación RAT desde ella misma le recomendamos que configure únicamente la interfaz principal (sin entrar dentro del menú **Options**), los siguientes valores:



- **Listen:** Activado (en verde)
- < **Master** >**:** Seleccionado con un volumen entre 60-70 (68, por ejemplo)
- **Talk:** Activado (en verde)
- < **Capture** >**:** Seleccionado con un volumen entre 60-70 (68, por ejemplo)
- Si tiene problemas para oír a un participante concreto y se ha asegurado de que ambos han configurado correctamente el audio, puede subir el volumen de dicho participante en la lista de participantes.

Si utiliza un fichero de configuración

Si desea configurar la aplicación RAT a través de su fichero de configuración le recomendamos los siguientes valores:

```
[node]
servicemanagers = servicemanager0
                  #O el nombre que tenga
                  #su configuracion

[servicemanager0]
url =   #Si el audio esta en una maquina
        #remota hay que indicarlo aqui con su IP

services = service_audio ...
           #Mas otros servicios que tenga en su
           #configuracion (service0 service1, etc.)

```

```
builtin = 1
name =

[service_audio]
packageName = AudioService.zip
serviceConfig = service_audioconfig0

[service_audioconfig0]
Force AC97 = False
Start Priority = 5

Output Gain = 68   #Un valor entre 60-70 (ej, 68)
Input Gain = 68    #Un valor entre 60-70 (ej, 68)

Position Window = Off
Use site id = On
Silence Suppression = Off
Talk = On
```

Se recomienda que lea la siguiente sección para una correcta configuración del sonido.

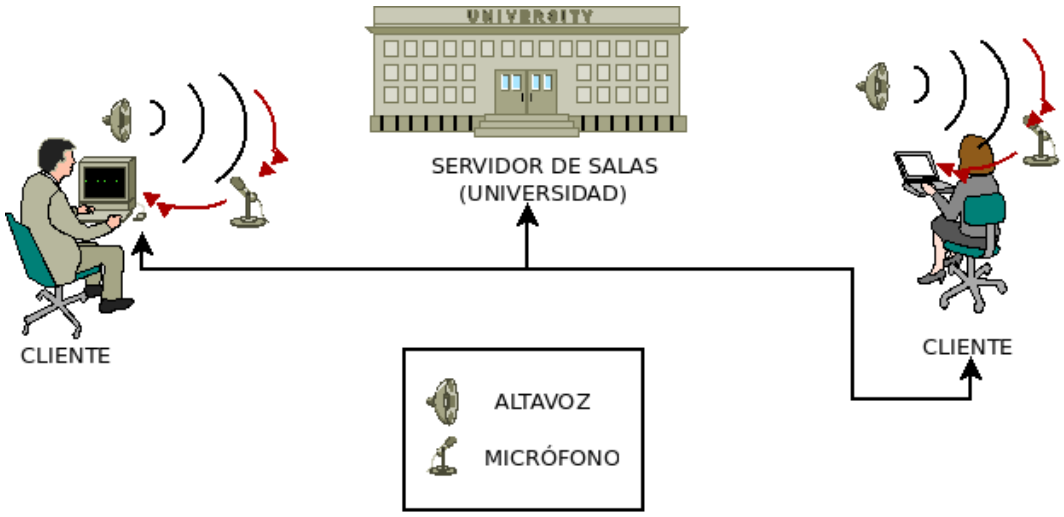
A.7.5. Problemas con el eco y soluciones

Como se ha comentado anteriormente, el sonido es una de las características clave en una videoconferencia, ya que será el medio más habitual que se utilizará para el intercambio de información entre los participantes.

Un problema muy común que existe en toda videoconferencia es el eco. El eco es el efecto que se produce con la **reflexión del sonido**. La señal acústica original ya se ha extinguido pero, sin embargo, aún devuelve sonido en forma de onda reflejada.

¿Cómo se produce el eco?

En nuestro caso, el eco se produce porque se producen reflexiones del sonido al viajar éste desde los altavoces a los micrófonos. Para entenderlo mejor, se puede observar la siguiente imagen, que es el ejemplo más simple en el que se puede producir eco en una videoconferencia:



En esta imagen podemos ver a dos participantes (dos *Clientes de Sala*) que se conectan al **Servidor de Salas** de una Universidad, a través de sus propios PCs personales de casa o trabajo (un PC, un portátil, etc.). Los elementos necesarios para poder realizar una videoconferencia son:

- Una cámara o webcam para transmitir el vídeo
- Unos altavoces para la recepción del sonido (para escuchar a los que nos hablan)
- Un micrófono para emitir sonido (para poder hablarles)

Teniendo esto en cuenta y, centrándonos en el sonido, el flujo normal de comunicación es el siguiente:

- Un participante emisor desea hablar a través de su micrófono
- El participante receptor escucha al participante emisor (recibe el sonido) a través de sus altavoces
- Ídem, pero en sentido contrario, para el participante receptor

En este flujo es donde ocurre el eco. Cuando el sonido llega al participante receptor, éste recibe el sonido a través de sus altavoces, pero, el sonido (como todas las ondas) son multidireccionales, por lo que, además de que el participante receptor escuche el sonido transmitido, dicho sonido viaja por toda la habitación o lugar donde se encuentre el participante. Por lo tanto, el mismo sonido recibido puede entrar en el micrófono del participante receptor, por lo que el sonido recibido puede ser devuelto al lugar donde se originó, aunque con menos intensidad que la señal original (se oye más bajo).

Este suceso se llama **realimentación del sonido**, que vuelve a ocurrir en el ordenador del participante emisor: vuelve a salir el sonido de los altavoces y vuelve a entrar por el micrófono (con menos volumen aún). Esto ocurre tantas veces hasta que el volumen del sonido se vuelve inapreciable, por lo que no entraría en ningún micrófono y no se repetiría. Evidentemente esto ocurriría en el sentido inverso, si el participante receptor se convierte en participante emisor.

La situación puede empeorar más aún si se multiplican el número de participantes (el sonido viajaría por cada uno de los altavoces de cada participante y entraría por cada micrófono de cada uno de ellos por lo que puede llegar a **distorsionar** el sonido). Es más, se puede empeorar más aún si un solo participante posee varias fuentes de emisión de sonido (múltiples micrófonos), porque el sonido que sale de los altavoces entraría por cada uno de los micrófonos.

Soluciones para reducir o eliminar el eco

Existen varias soluciones para poder reducir o eliminar completamente el eco en una reunión o videoconferencia. La solución más económica sería utilizar, en lugar de unos altavoces para escuchar a los participantes, unos auriculares. Así, el sonido sale de los auriculares, pero no se realimentan de nuevo en los micrófonos. Esta solución es más adecuada para participantes que estén conectados a la Sala a través de su PC personal o un portátil. Además, si no dispone de micrófono, es más aconsejable utilizar unos auriculares con micrófono incorporado, diseñado para comunicaciones a través de videoconferencia.

Esta solución también se podría plantear en una sala con varios participantes (como en la imagen anterior), pero sería poco práctico y, además, un poco incómodo.

Para estas salas, la solución aconsejada es la de instalar un **Cancelador de Eco**. El cancelador de eco es un componente hardware en el que se le conectan todas las fuentes de audio que vayan a transmitir (los micrófonos en nuestro caso), éste procesa todas las señales de audio que reciba o vaya a emitir mediante una configuración establecida, buscando eco para eliminarlas, y envía o recibe la señal procesada eliminando todo eco que detecte.

El cancelador de eco instalado en el Aula de Teledocencia es un **Nexia** de la marca **Biamp Professional Audio Systems**. Éste se conecta a un PC a través de un cable Ethernet y, utilizando un software propietario de la casa Nexia, nos permite configurarlo a nuestro gusto.

La configuración del cancelador de eco no es sencilla, existen profesionales en el sector audiovisual en el que uno de sus trabajos consiste en configurar correctamente el cancelador de eco en el lugar donde

se encuentre instalado. Por ello no nos es posible explicar cómo se puede configurar el cancelador de eco.

Con dicho cancelador de eco, se ha configurado para que detecte tanto los micrófonos de la Sala, como el de petaca, así como el micrófono que está en el Atril. Éste detecta todo patrón de sonido que se repita (el eco), ”corta” esta señal para eliminar el eco y envía el sonido cortado (sin eco).

El inconveniente de este método es que se trata de una solución bastante costosa: por un lado hay que comprar el cancelador de eco y, por otro, hay que contratar los servicios de un profesional cualificado para configurar correctamente el cancelador según nuestras necesidades. Sin embargo, es la solución más profesional y cómoda.

Por último, existen soluciones intermedias, en cuanto a coste y calidad de sonido. Por ejemplo, existen sets de microfonía para conferencias en las que se incluyen varios micrófonos en un solo aparato que se conecta, normalmente, por USB. Además, algunos de estos aparatos incluyen su propio altavoz y cancelador de eco por lo que no haría falta configurar nada, simplemente lo básico para ser reconocido por el PC al que se le vaya a conectar. Un ejemplo de dicho aparato es el **YAMAHA PJP-25UR**:

Esta solución es recomendable para aquellas salas de reuniones en las que hay pocos asistentes. Como se ha comentado, las ventajas de dicha solución es que suelen incluir cancelador de eco, sin necesidad de configuración alguna y su coste es medio en comparación con la solución anterior. Además, como ya se ha comentado, es apto para pequeñas salas, o salas de reuniones en las que hay pocos participantes en la sesión.

A.8. Guardar la Configuración

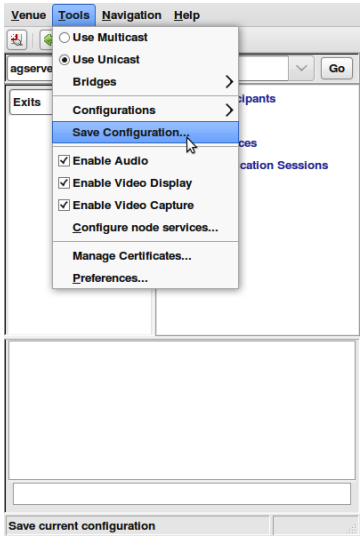
Como se ha comentado varias veces en secciones anteriores, puede guardar qué servicios quiere que se ejecuten al entrar en una sala, así como la configuración de cada uno de ellos. De esta forma, se ahorra tener que repetir el mismo procedimiento cada vez que entre en una sala. A continuación se explicará los pasos a seguir para guardar una configuración para luego cargarla.

Por otro lado, estas configuraciones almacenadas, lo que realmente hace Access Grid es almacenar cada configuración en un fichero, almacenado en una carpeta del Cliente, luego, para cargar la configuración, lee ese fichero para aplicar los cambios. Así que, para usuarios más experimentados o para aquellos que tengan algún problema en almacenar la configuración, se explicará brevemente cómo crear un fichero de configuración personalizado desde un editor de textos.

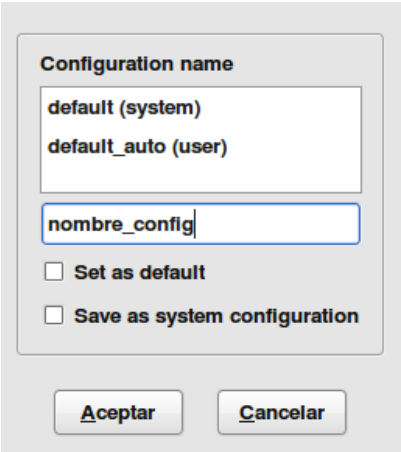
A.8.1. Guardar la configuración desde el Cliente de Access Grid

Una vez haya finalizado de añadir los servicios y configurar cada uno de ellos, haga lo siguiente para almacenar la configuración:

- En la ventana del Cliente de Access Grid, diríjase a *Tools* $->$ *Save Configuration...*:



- Le aparecerá la siguiente ventana, en ella puede observar un listado de configuraciones ya almacenadas (default, etc.) y un cuadro de texto para introducir el nombre de la configuración que queremos almacenar:



- Una vez introducido el nombre, si lo desea, puede marcar la opción **Set as default**, de este modo, cada vez que ejecute el Cliente de Access Grid, cargará esta configuración por defecto.

- Si marca la opción **Save as system configuration** la configuración será almacenada para ser usado por todos los usuarios que accedan al sistema
- Pulse el botón **Aceptar** para finalizar.

A.8.2. Cómo crear un fichero de configuración

Para aquellos usuarios experimentados que quieran crear sus ficheros de configuración personalizados, o para aquellos que, por algún motivo, tenga algún tipo de problema que le impida guardar la configuración deseada, se explicará en esta sección cómo crear su propio fichero de configuración.

Antes de nada, hay que saber dónde guardar dicho fichero de configuración, dependiendo del Sistema Operativo, la ruta de almacenamiento es:

Ruta de Configuraciones de Usuario:

- **Windows:** `C:/Archivos de Programa/AGTk-3/config/nodeConfig`
- **Linux:** `/home/ usuario-del-sistema /.AccessGrid3/Config/nodeConfig`

Nota: *Como puede comprobar, en Linux, el directorio .AccessGrid3 es una carpeta oculta, por lo que deberá activar la opción de Mostrar los archivos ocultos*

Ruta de Configuraciones del Sistema:

- **Windows:** `C:/Documents and Settings/ usuario-del-sistema /Datos de Programa/AccessGrid3/Config/nodeConfig`
- **Linux:** `/etc/AccessGrid3/Config/nodeConfig`

A.8.3. Estructura del fichero de configuración

Un ejemplo, sencillo y básico, es el fichero de configuración que trae por defecto Access Grid, puede visualizarlo desde aquí:

```
# AGTk 3.2 node configuration
[node]
servicemanagers = servicemanager0

[serviceconfig2]
Position Window = Justify Left
Start Priority = 7
Thumbnail Columns = 2

[resource0]
name = V4L2:/dev/video0

[service2]
packageName = VideoConsumerService.zip
serviceConfig = serviceconfig2

[service1]
packageName = AudioService.zip
serviceConfig = serviceconfig1

```

```
[service0]
packageName = VideoProducerService.zip
resource = resource0
serviceConfig = serviceconfig0

[servicemanager0]
url =
services = service0 service1 service2
builtin = 1
name =

[serviceconfig1]
Force AC97 = False
Start Priority = 5
Output Gain = 68
Position Window = Off
Input Gain = 68
Use site id = On
Silence Suppression = Off
Talk = On

[serviceconfig0]
Frame Rate = 25
Encoding = h261
Start Priority = 5
Standard = PAL
Bandwidth = 800
Port = zc3xx
Quality = 80
Stream Name = V4L2:/dev/video0
```

Dicho fichero podemos ver que está separado en **secciones escritas entre corchetes** [ ]. Por otro lado, cada sección, contiene una serie de opciones. La estructura de las opciones es:

```
nombre_opcion = valor_opcion
```

Las secciones que podemos encontrar son:

- **[node]**: En esta sección se escriben los servicesmanager que tendrá nuestro fichero de configuración, su sintaxis es:

```
servicemanagers = nombre-servicemanager
```

  Donde *nombre-servicemanager* es el nombre que queremos darle a nuestro *ServiceManager*, dicho **nombre debe aparecer luego en el fichero de configuración como una sección más**. Es decir, debe existir una sección llamada [*nombre-servicemanager*]

- **[nombre-servicemanager]** (Depende del nombre escrito en el apartado servicemanagers de la sección [node]): En esta sección indicamos los parámetros relacionados con el *ServiceManager*. Éstos son:

  - **url:** URL donde esté ejecutándose el *ServiceManager* (ver *Añadir Servicios situados en otra máquina*). Si la URL está vacía, estamos indicando que no usamos ningún ServiceManager,

salvo el nuestro local (que se ejecuta siempre). Es decir, indicamos que, únicamente, añadiremos servicios locales.

- **services = nombre-servicio1 nombre-servicio2 ...:** Indicamos los servicios que queremos utilizar en este *ServiceManager*. Los servicios se indican por su nombre de sección, es decir, debe de existir, en el fichero de configuración, las secciones [*nombre-servicio1*], [*nombre-servicio2*], etc., tantos como servicios queramos añadir.
- **builtin:** Por confirmar Por defecto siempre está a 1.
- **name:** Para darle un nombre a este *ServiceManager*.

- **[nombre-servicio] (service1, service2, etc. Depende del nombre escrito en el apartado services de la sección [nombre-servicemanager]):** Son los servicios indicados en la sección anterior. Dentro de esta sección podemos encontrar:

  - **packageName:** Fichero .zip del servicio que queremos añadir. Por ejemplo, si queremos añadir un VideoService, el valor seria *VideoService.zip*.
  - **resource (sólo servicios de vídeo):** Indicamos el recurso, es decir, la fuente de vídeo, que queremos utilizar en este servicio. Dicho recurso se indica por su nombre de sección, es decir, si escribimos el valor resource1 debe de existir, en el fichero de configuración, una sección llamada [*resource1*].
  - **serviceConfig:** Indicamos la configuración del servicio que queremos utilizar. Dicha configuración se indica por el nombre de sección, es decir, si escribimos el valor *serviceconfig1*, debe de existir, en el fichero de configuración, una sección llamada [*serviceconfig1*]

- **[nombre-resource] (resource1, resource2, etc. Depende del nombre escrito en el apartado resource de la sección [nombre-servicio]):** En esta sección indicamos el nombre de la fuente de vídeo a enlazar con resourceX. Dentro de esta sección podemos encontrar:

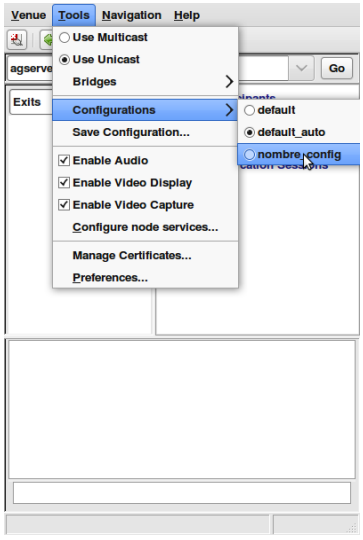
  - **name:** Dirección del recurso/fuente de vídeo. Por ejemplo, en Linux, podría ser `V4L2¬ :/dev/video0`; sin embargo, en Windows, sería el nombre del dispositivo, como por ejemplo, **Microsoft WDM Image Capture (Win32)**.

- **[nombre-serviceconfig] (serviceconfig1, serviceconfig2, etc. Depende del nombre escrito en el apartado _serviceConfig de la sección [nombre-servicio]):** Esta sección es completamente dependiente del tipo de servicio añadido, ya que, si es un servicio de Audio, tendrá unos apartados diferentes a los de un servicio de Video, de Escritorio Compartido, etc. Explicaremos, de momento, los más usuales que son los servicios de Audio y Vídeo:

  - **Audio:** Si el tipo de servicio es de Audio, podemos encontrar los siguientes apartados. Si no sabe para qué sirve cada uno de estos parámetros visite la sección *Configuración de RAT*:
    - **Force AC97** [*False/True*]
    - **StartPriority** [*1-10*]
    - **Output Gain** [*0-100*]
    - **Position Window** [*Off/On*]
    - **Input Gain** [*0-100*]
    - **Use site id** [*Off/On*]
    - **Silence Suppression** [*Off/On*]
    - **Talk** [*Off/On*]

- **Video:** Si el tipo de servicio es de Vídeo, dependiendo del tipo de servicio de Video, si es **Consumidor** o **Productor**, o si utiliza H.264/MPEG-4 en lugar de H.261, podemos encontrar unos apartados u otros. Se detalla los más comunes. Si no sabe para qué sirve cada uno de estos parámetros visite la sección *Configuración y valores óptimos de Vídeo*:
  - **Mute Sources** [*Off/On*]
  - **Thumbnail Columns** [*1-10*]
  - **Frame Rate** [*1-30*]
  - **Encoding** [*h261-mpeg4-h264-...*]
  - **Start Priority** [*1-10*]
  - **Transmit on Startup** [*Off-On*]
  - **Position Window** [*Off/Justify Left/Justify Right*]
  - **Standard** [*NTSC-PAL*]
  - **Bandwidth** [*1-3000*]
  - **Port** [*puerto*]
  - **Quality** [*1-100*]
  - **Stream Name** [*ruta (ej* `V4L2:/dev/video0`*)*]

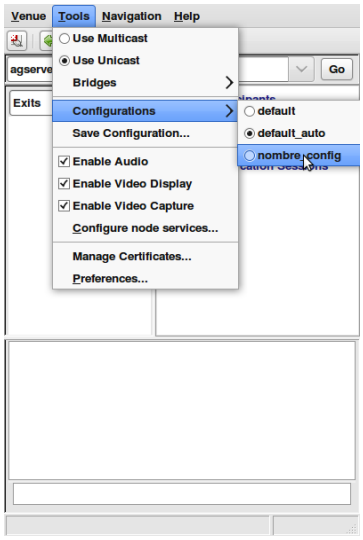
Una vez terminado de realizar el fichero de configuración, basta con guardarlo. Ejecute el cliente de Access Grid, y verá, en Tools ->Configurations, el nombre del fichero de configuración creado.



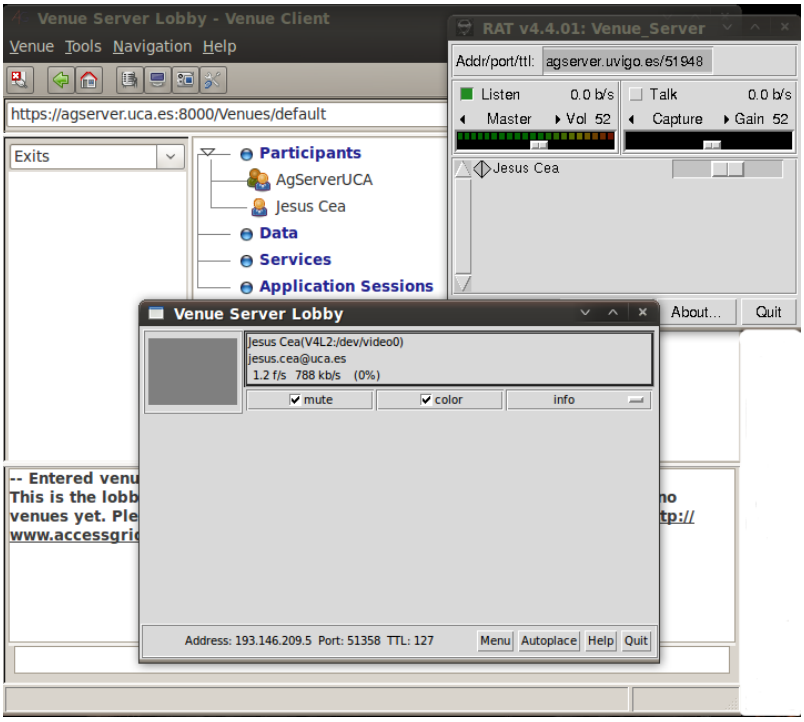
A.8.4. Cargar un perfil de configuración

Para cargar una configuración determinada haga lo siguiente:

- En la ventana del Cliente de Access Grid, diríjase a *Tools* $->$ *Configurations*:



- Aparecerá un menú desplegable con todas las configuraciones almacenadas en su sistema. Seleccione la configuración que desea cargar. Automáticamente comenzará a cargar los servicios establecidos en dicha configuración.

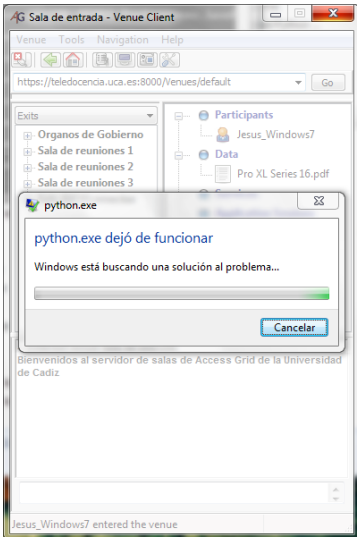


A.9. Problemas y Soluciones

A.9.1. Access Grid en Windows 7

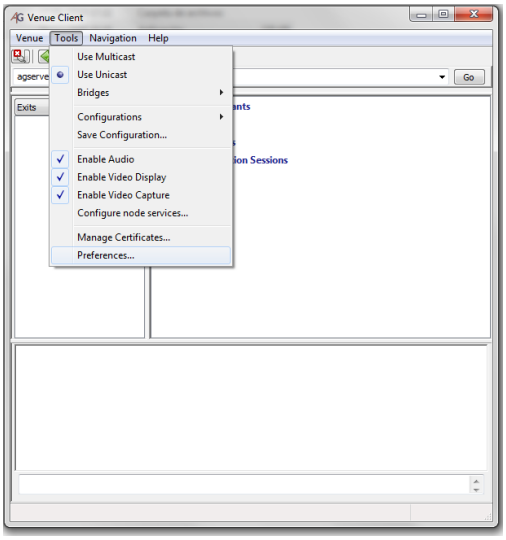
Access Grid en Windows 7 funciona correctamente, salvo por un error producido por el cliente Beacon integrado en el Cliente. Dicho error, una vez se entre en una sala, produce un error del tipo *"Python*

dejó de funcionar", bloqueando la aplicación y, una vez se acepte el mensaje de error, se cierra la aplicación.

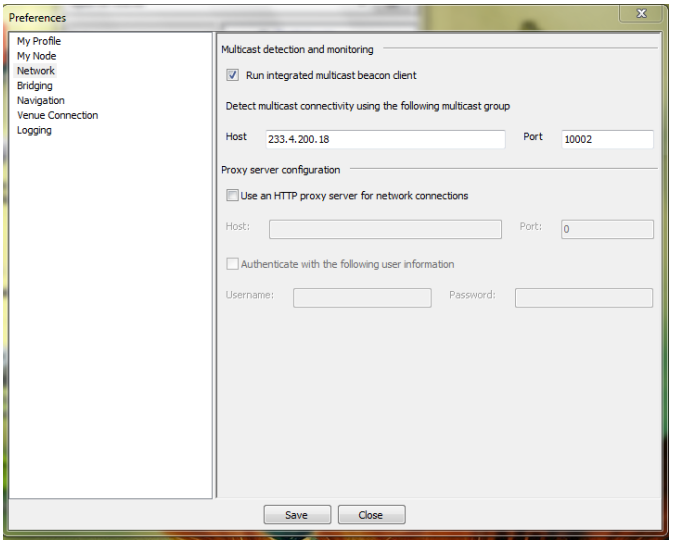


Para solucionar este problema hay que desactivar el cliente de Beacon integrado. Para ello:

- En el cliente de Access Grid, diríjase a *Tools* $->$ *Preferences*:



- Nos aparecerá la ventana de configuración del Cliente de Access Grid:



- Nos vamos a la pestaña *Network* y desactivamos la opción **"Run integrated multicast beacon client"**:

- Guardamos los cambios pulsando el botón **Save**

A.10. Enlaces de Interés

Página Oficial: http://www.accessgrid.org/

Página Oficial, sitio para desarrolladores: http://www.accessgrid.org/developer

VisLab. Un grupo de la Universidad de QueensLand (Australia) especializado en visualización avanzada, ciencias de la computación y colaboran en otros proyectos/tecnologías, entre ellas, Access Grid.:
http://www.vislab.uq.edu.au/research/accessgrid/background.html

Ja.net. Contiene un apartado interesante sobre Access Grid, sobre todo porque exponen casos de estudio bajo Access Grid:
http://www.ja.net/services/video/agsc/AGSCHome/index.html http://www.ja.net/services/video/agsc/services/service
Universidad de los Andes: Proyecto bajo Access Grid hecho por la Universidad de los Andes donde podemos ver proyectos interesantes para este sistema, como la aplicación **SharedPaint** (una pizarra compartida) o **ShareVTK** para el desarrollo colaborativo de diseños en 3D: http://ag-mox.uniandes.edu.co/website/es/ind

Enlaces para desarrolladores:
http://kenai.com/projects/sharedpaint/sources/AGTk-SharedPaint/show http://kenai.com/projects/sharedpaint/downlo
http://sharedpaint.kenai.com/
Control de Acceso de las Salas

OpenSSL: http://www.openssl.org/docs/apps/x509.html **Tutorial OpenSSL:** http://helektron.com/tutorial-como-crear-una-autoridad-certificadora-ca-con-openssl/

Apéndice B

Scripts de Arranque del sistema *Access Grid*

B.1. Fichero de configuración del Servidor de Salas

```
# Nombre del ejecutable de VenueServer. Normalmente es
# /usr/bin/VenueServer3.py, sin embargo en sistemas Fedora
# y otros sistemas basados en rpm usan /usr/bin/VenueServer
AGSERVER_EXE=/usr/bin/VenueServer3.py
#
# Login del usuario del sistema (el servidor se ejecutara bajo su propiedad
   )
AGSERVER_OWNER=uca
#
# Localizacion del directorio donde quiere que se ejecute el servidor
AGSERVER_HOME=/home/uca
#
# Nombre del fichero de bloqueo del VenueServer
AGSERVER_LOCKFILE=/var/lock/agvs
#
```

B.2. Demonio de arranque del Servidor de Salas

```
#!/bin/bash
#
# agvs        Script de arranque para Access Grid Venue Server
#
# chkconfig: - 84 16
# description: Venue server para Access Grid 3.xx
#
# config: /etc/default/agvs.conf


if [ -f /etc/default/agvs.conf ]; then
        . /etc/default/agvs.conf
fi

```

```
15  [ -z ${AGSERVER_OWNER} ] && {
16    echo "AGSERVER_OWNER no esta establecido. Establezcalo en /etc/default/
        agvs.conf"
17    exit 1
18  }
19  [ -z ${AGSERVER_HOME} ] && {
20    echo "AGSERVER_HOME no esta establecido. Establezcalo /etc/default/agvs.
        conf"
21    exit 2
22  }
23  [ -z ${AGSERVER_EXE} ] && {
24    echo "AGSERVER_EXE no esta establecido. Establezcalo /etc/default/agvs.
        conf"
25    exit 1
26  }
27
28  # Path to the venue server script and lock file.
29  prog=VenueServer
30  lockfile=${AGSERVER_LOCKFILE:-/var/lock/agvs}
31  RETVAL=0
32
33  [ -x ${AGSERVER_EXE} ] || {
34    echo "${AGSERVER_EXE} debe existir y ser ejecutable (permisos de
        ejecucion)"
35    exit 3
36  }
37
38
39  start() {
40          echo -n $"Iniciando $prog: "
41          (cd ${AGSERVER_HOME} && su ${AGSERVER_OWNER} -c ${AGSERVER_EXE} &)
42          RETVAL=$?
43          echo
44          [ $RETVAL = 0 ] && touch ${lockfile}
45          return $RETVAL
46  }
47
48  stop() {
49          echo -n $"Deteniendo $prog ... "
50          pkill -HUP ${prog}
51          RETVAL=$?
52          [ $RETVAL -ne 0 ] && {
53            sleep 3
54            echo -n $"... "
55            pkill -TERM ${prog}
56          }
57          [ $RETVAL -ne 0 ] && {
58            sleep 3
59            echo -n $"... "
60            pkill -KILL ${prog}
61          }
62          RETVAL=$?
63          echo
64          [ $RETVAL = 0 ] && rm -f ${lockfile}
65  }
66
```

```
# Llamadas
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        stop
        start
        ;;
  condrestart)
        if [ -f ${pidfile} ] ; then
                stop
                start
        fi
        ;;
  *)
        echo $"Forma de uso: $prog {start|stop|restart|condrestart}"
        exit 1
esac

exit $RETVAL
```

B.3. Fichero de configuración de Registry

```
# Nombre del ejecutable de Registry Peer. Normalmente suele ser
# /usr/bin/RegistryPeer3.py, sin embargo, en sistemas basados
# en RPM (como Fedora) suele ser en /usr/bin/RegistryPeer
AGREGISTRY_EXE=/usr/bin/RegistryPeer3.py
#
# Login del usuario (del sistema) que desea iniciar el registry peer
AGREGISTRY_OWNER=uca
#
# Ruta Home del usuario anterior
AGREGISTRY_HOME=/home/uca
#
# Servidor web o fichero asociado para el Registry peer
PEERLISTURL=teledocencia.uca.es
#
# Ruta de fichero de bloqueo de Registry Peer
AGREGISTRY_LOCKFILE=/var/lock/agregistry
#
```

B.4. Demonio de arranque de Registry

```
#!/bin/bash
#
# agregistry        Script de arranque para Access Grid Registry Peer
```

```
#
# chkconfig: - 85 15
# description: Registry Peer para Access Grid 3.xx
#
# config: /etc/default/agregistry.conf


if [ -f /etc/default/agregistry.conf ]; then
        . /etc/default/agregistry.conf
fi

[ -z ${AGREGISTRY_EXE} ] && {
  echo "AGREGISTRY_EXE no esta establecido. Establezcalo en /etc/default/
     agregistry.conf"
  exit 1
}
[ -z ${AGREGISTRY_OWNER} ] && {
  echo "AGREGISTRY_OWNER no esta establecido. Establezcalo en /etc/default/
     agregistry.conf"
  exit 1
}
[ -z ${AGREGISTRY_HOME} ] && {
  echo "AGREGISTRY_HOME no esta establecido. Establezcalo en /etc/default/
     agregistry.conf"
  exit 2
}
[ -z ${PEERLISTURL} ] && {
  echo "PEERLISTURL no esta establecido. Establezcalo en /etc/default/
     agregistry.conf"
  exit 3
}

# Rutas del script registry peer, logs y fichero de bloqueo.
prog=RegistryPeer
rp_stdout_log=${AGREGISTRY_HOME}/.AccessGrid3/Logs/RegistryPeerStdOut.log
rp_stderr_log=${AGREGISTRY_HOME}/.AccessGrid3/Logs/RegistryPeerStdErr.log
lockfile=${AGREGISTRY_LOCKFILE:-/var/lock/agregistry}
RETVAL=0

[ -x ${AGREGISTRY_EXE} ] || {
  echo "${AGREGISTRY_EXE} debe existir y ser ejecutable (permisos de
     ejecucion)"
  exit 4
}


start() {
    echo -n "Iniciando $prog ..."
    cd ${AGREGISTRY_HOME} && su ${AGREGISTRY_OWNER} \
      -c "${AGREGISTRY_EXE} \
      -u ${PEERLISTURL} \
      1>>${rp_stdout_log} \
      2>>${rp_stderr_log}" &
    RETVAL=$?
    [ $RETVAL = 0 ] && touch ${lockfile}
    echo
```

```
}

stop() {
        echo -n $"Deteniendo $prog ... "
        pkill -HUP ${prog}
        RETVAL=$?
        [ $RETVAL = 0 ] && rm -f ${lockfile}
        echo
}

# Llamadas.
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        stop
        sleep 2
        start
        ;;
  *)
        echo $"Formas de uso: $prog {start|stop|restart}"
        exit 1
esac

exit $RETVAL
```

B.5. Fichero de configuración de Bridge Server

```
# Ruta completa del ejecutable del Bridge Server. Normalmente esta
# en /usr/bin/Bridge3.py, pero en sistemas basados en RPM (Fedora, etc)
# suele ser /usr/bin/Bridge
AGBRIDGE_EXE=/usr/bin/Bridge3.py
#
# Login del usuario que desea ejecutar el bridge
AGBRIDGE_OWNER=uca
#
# Ruta Home del usuario anterior
AGBRIDGE_HOME=/home/uca
#
# Ruta del fichero de bloqueo del Bridge Server
AGBRIDGE_LOCKFILE=/var/lock/agbs
#
# Servidor Web o fichero asociado a este bridge (teledocencia.uca.es)
PEERLISTURL=teledocencia.uca.es
#
# Un nombre corto identificando este Bridge (ej "UCA-PuertoReal")
AGBRIDGE_NAME=UCA-PuertoReal
#
```

```
# Una cadena indicando la localizacion de este Bridge Server ("Cadiz, Spain
    ")
AGBRIDGE_LOCATION="Cadiz, Spain"
#
```

B.6. Demonio de arranque de Bridge Server

```
#!/bin/bash
#
# agbs        Script de arranque para Access Grid Bridge Server
#
# chkconfig: - 86 14
# description: Servicio Bridge para Access Grid 3.xx
#
# config: /etc/default/agbs.conf
CONFIGFILE=/etc/default/agbs.conf


if [ -f ${CONFIGFILE} ]; then
        . ${CONFIGFILE}
fi

[ -z ${AGBRIDGE_OWNER} ] && {
  echo "AGBRIDGE_OWNER no esta establecido. Establezcalo en ${CONFIGFILE}"
  exit 1
}
[ -z ${AGBRIDGE_HOME} ] && {
  echo "AGBRIDGE_HOME no esta establecido. Establezcalo en ${CONFIGFILE}"
  exit 2
}
[ -z ${AGBRIDGE_EXE} ] && {
  echo "AGBRIDGE_EXE no esta establecido. Establezcalo en ${CONFIGFILE}"
  exit 1
}
[ -z ${AGBRIDGE_NAME} ] && {
  echo "AGBRIDGE_NAME no esta establecido. Establezcalo en ${CONFIGFILE}"
  exit 1
}
[ -z "${AGBRIDGE_LOCATION}" ] && {
  echo "AGBRIDGE_LOCATION no esta establecido. Establezcalo en ${CONFIGFILE
      }"
  exit 1
}
[ -z ${PEERLISTURL} ] && {
  echo "PEERLISTURL no esta establecido. Establezcalo en ${CONFIGFILE}"
  exit 1
}

prog=Bridge
bs_stdout_log=${AGBRIDGE_HOME}/BridgeStdOut.log
bs_stderr_log=${AGBRIDGE_HOME}/BridgeStdErr.log
lockfile=${AGBRIDGE_LOCKFILE:-/var/lock/agbs}
RETVAL=0
```

```
46
47  [ -x ${AGBRIDGE_EXE} ] || {
48    echo "${AGBRIDGE_EXE} debe existir y ser ejecutable (permisos de
        ejecucion)"
49    exit 3
50  }
51
52
53  start() {
54          echo -n $"Iniciando $prog: "
55          (cd ${AGBRIDGE_HOME} && su ${AGBRIDGE_OWNER} \
56            -c "${AGBRIDGE_EXE} \
57            -u ${PEERLISTURL} \
58            -n ${AGBRIDGE_NAME} \
59            -l ${AGBRIDGE_LOCATION} \
60            1>>${bs_stdout_log} 2>>${bs_stderr_log}" &)
61          RETVAL=$?
62          echo
63          [ $RETVAL = 0 ] && touch ${lockfile}
64          return $RETVAL
65  }
66
67  stop() {
68          echo -n $"Deteniendo $prog ... "
69          pkill -HUP ${prog}
70          RETVAL=$?
71          [ $RETVAL -ne 0 ] && {
72            sleep 3
73            echo -n $"... "
74            pkill -TERM ${prog}
75          }
76          [ $RETVAL -ne 0 ] && {
77            sleep 3
78            echo -n $"... "
79            pkill -KILL ${prog}
80          }
81          RETVAL=$?
82          echo
83          [ $RETVAL = 0 ] && rm -f ${lockfile}
84  }
85
86  # Llamadas.
87  case "$1" in
88    start)
89          start
90          ;;
91    stop)
92          stop
93          ;;
94    restart)
95          stop
96          start
97          ;;
98    *)
99          echo $"Formas de uso: $prog {start|stop|restart}"
100         exit 1
```

```
101  esac
102
103  exit $RETVAL
```

Apéndice C

Software y Librerías Usadas

A continuación se detalla todas las herramientas que han sido necesarias utilizar para la realización de este Proyecto Final de Carrera.

C.1. Python 2.4

Access Grid está desarrollado en *Python 2.4*, así que, tanto para mejorar y corregir el código del propio sistema, como para el desarrollo de Aplicaciones Compartidas, se tuvo que realizar en *Python 2.4* [1].

Figura C.1: Logo de Python

Python es un **lenguaje de programación de alto nivel**, creado por Guido van Rossum. Su filosofía hace hincapié en una sintaxis muy limpia, favoreciendo un código legible. Se trata de un **lenguaje de programación multiparadigma** ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Python es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y es multiplataforma.

C.1.1. Filosofía

Los usuarios de Python se refieren a menudo a la **Filosofía Python**, que es bastante análoga a la filosofía de Unix. El código que sigue los principios de Python de legibilidad y transparencia se dice que es ”pythonico”. Contrariamente, el código opaco u ofuscado es bautizado como ”no pythonico” (”unpythonic” en inglés). Estos principios fueron famosamente descritos por el desarrollador de Python Tim Peters en *El Zen de Python*:

- Bello es mejor que feo.
- Explícito es mejor que implícito.

[1] `http://www.python.org/download/releases/2.4/`

- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Ralo es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son tan especiales como para quebrantar las reglas.
- Aunque lo práctico gana a la pureza.
- Los errores nunca deberían dejarse pasar silenciosamente.
- A menos que hayan sido silenciados explícitamente.
- Frente a la ambigüedad, rechaza la tentación de adivinar.
- Debería haber una -y preferiblemente sólo una- manera obvia de hacerlo.
- Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.
- Ahora es mejor que nunca.
- Aunque nunca es a menudo mejor que ya mismo.
- Si la implementación es difícil de explicar, es una mala idea.
- Si la implementación es fácil de explicar, puede que sea una buena idea.
- Los espacios de nombres (namespaces) son una gran idea ¡Hagamos más de esas cosas!.

A modo de curiosidad, desde la versión 2.1.2, *Python* incluye estos puntos (en su versión original en inglés) como un **huevo de pascua** que se muestra al ejecutar *import this*.

C.1.2. Características de Python

Propósito general

Se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.

Multiplataforma

Hay versiones disponibles de Python en multitud de Sistemas Operativos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

Interpretado

Quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos *bytecodes* que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.

Interactivo

Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como también para los programadores más avanzados: se pueden probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa.

Existen otros programas, tales como *IDLE*, *bpython* o *IPython*, que añaden funcionalidades extra al modo interactivo, como el auto-completado de código y el coloreado de la sintaxis del lenguaje.

Orientado a Objetos

La programación orientada a objetos está soportada en *Python* y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

Funciones y librerías

Python tiene una gran biblioteca estándar, usada para una diversidad de tareas. Esto viene de la filosofía "pilas incluidas" ("batteries included") en referencia a los módulos de *Python*. Los módulos de la biblioteca estándar pueden mejorarse por módulos personalizados escritos tanto en *C* como en *Python*. Debido a la gran variedad de herramientas incluidas en la biblioteca estándar, combinada con la habilidad de usar lenguajes de bajo nivel como *C* y *C++*, los cuales son capaces de interactuar con otras bibliotecas, *Python* es un lenguaje que combina su clara sintaxis con el inmenso poder de lenguajes menos elegantes.

Sintaxis clara

Por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave *begin* y *end*. Para separar las porciones de código en *Python* se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar. Además, se utilizan palabras en inglés donde otros lenguajes utilizarían símbolos (por ejemplo, los operadores lógicos !, || y && en Python se escriben *not, or y and*, respectivamente).

C.1.3. Ejemplo de Python

```
def factorial(x):
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)
```

C.2. wxPython

wxPython es una colección de *bindings* de la biblioteca gráfica ***wxWidgets***. *wxWidgets* es una librería gráfica escrita en *C++* que permite a los desarrolladores crear aplicaciones para *Linux*, *Windows* y *OS*

X, tanto en 32 bits como en 64 bits.

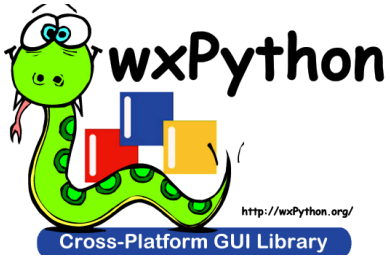


Figura C.2: Logo de wxPython

Están publicadas bajo una licencia **LGPL**, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste de licencias.

La biblioteca *wxWidgets* proporciona una interfaz gráfica basada en las bibliotecas ya existentes en el Sistema Operativo (nativas), con lo que se integran de forma óptima y resultan muy portables entre distintos Sistemas Operativos.

La interfaz de *Access Grid* está desarrollada bajo *wxPython*.

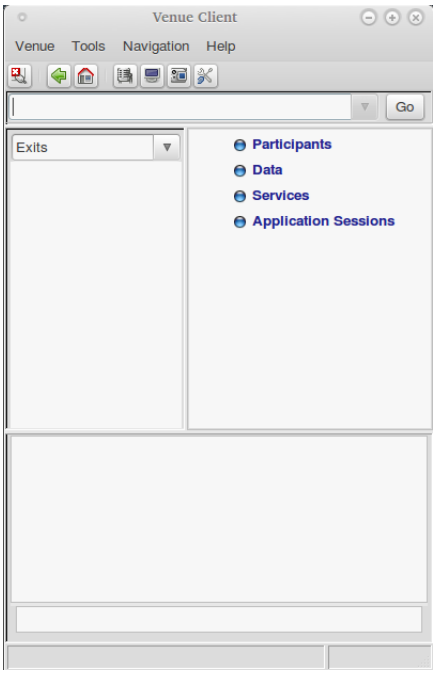


Figura C.3: Ejemplo de interfaz de wxPython

C.2.1. Características de wxPython

Multiplataforma

wxWidgets te permite orientar aplicaciones al estilo de Windows, Linux/Unix con las herramientas de GTK+ (o X11 o Motif) y MacOS. Con lo cual, no tienes porque preocuparte si la inversión en el

desarrollo será obsoleta debido a los requisitos de otras plataformas. Especificaciones de plataformas con muy poco poder son sacrificadas por la amplitud de cobertura, ya que la falta de funcionalidades es a menudo emulada, como en los marcos MDI(Interfaz de múltiples documentos) y los avanzados control de árboles y de listas. Si quieres, puedes incluso compilar aplicaciones Windows sin dejar tu entorno Linux.

Código abierto

wxWidgets es software **libre y abierto**. Está desarrollado por un grupo de entusiastas y con dotes quienes están construyendo unas herramientas de calidad las cuales hacen honor a sus propias expectativas y estándares.

Amplia Gama de controles

wxWidgets tiene controles básicos usuales tales como *cajas de texto, botones con imágenes, entrada de texto, listas con barras de desplazamiento, cuadro de seleccionador de ítemes, cuadros de selección*, y clases avanzadas como por ejemplo, *wxToolbar* para añadir una barra de botones, o *wxGrid* para implementar una tabla / cuadrícula.

Potente sistema de gestión de eventos

wxWidgets tiene un sistema de eventos similar a los mapas de mensaje de MFC (aunque más elegante y poderoso), que permite que los **eventos** se asocien con **funciones miembro** de manera estática (en tiempo de compilación) o dinámica (en tiempo de ejecución). A diferencia de la herramienta QT, esto no requiere un preprocesador no estándar.

Otras características

- Soporte de **bases de datos**
- Programación **Multihilo**
- Programación en red
- Soporte OpenGL
- etc.

C.3. PyQt4

Al igual que *wxPython*, *PyQt4* es una colección de *bindings* de la biblioteca gráfica ***Qt4***. *Qt4* es ampliamente conocido y extendido, tales como por ejemplo, el escritorio *KDE*, la aplicación *Google Earth* o la aplicación *Virtual Box* para instalar máquinas virtuales en su equipo.

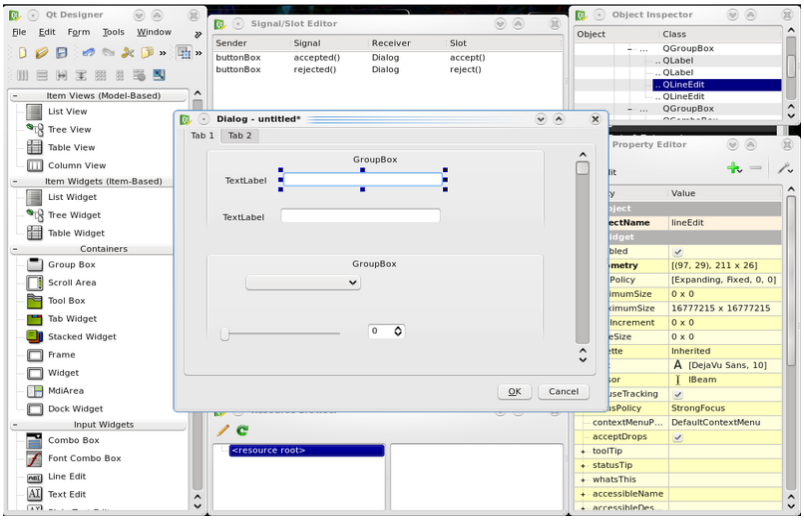


Figura C.4: QtCreator para diseñar interfaces Qt

Funciona en todas las principales plataformas, y tiene un amplio apoyo. Alguna de sus características son:

- Soporte de **bases de datos** gracias al módulo *SQL*
- Programación **Multihilo**
- Programación en red
- Soporte OpenGL
- Motor de scripting
- Soporte para la lectura y escritura de ficheros XML
- etc.

C.4. Subversion

Subversion es un sistema de **control de versiones** diseñado específicamente para reemplazar al popular *CVS*, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo *Apache/BSD* y se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de órdenes.

A diferencia de *CVS*, *Subversion* numera todo el repositorio con un único número identificando un estado común de todos los archivos del repositorio en un instante determinado.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración.

Las principales características de SVN y sus mejoras frente a CVS son:

- Mantiene versiones no sólo de archivos, sino también de directorios.

- También se mantienen versiones de los metadatos asociados a los directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Posibilidad de elegir el protocolo de red. Además de un protocolo propio (svn), puede trabajar sobre http (o https) mediante las extensiones *WebDAV*. *WebDAV* (más conocido como DAV) es un protocolo que amplia las posibilidades del HTTP/1.1 añadiendo nuevos métodos y cabeceras. La capacidad de funcionar con un protocolo tan universal como el http simplifica la implantación (cualquier infraestructura de red actual soporta dicho protocolo) y universaliza las posibilidades de acceso (si se quiere, puede utilizarse a través de Internet).
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Mayor eficiencia en la creación de ramas y etiquetas que en CVS, siguiendo un Orden Constante (O(1)).

C.5. Redmine

Durante todo el desarrollo de este Proyecto Final de Carrera se ha ido utilizando la aplicación web para gestión de proyectos *Redmine*.

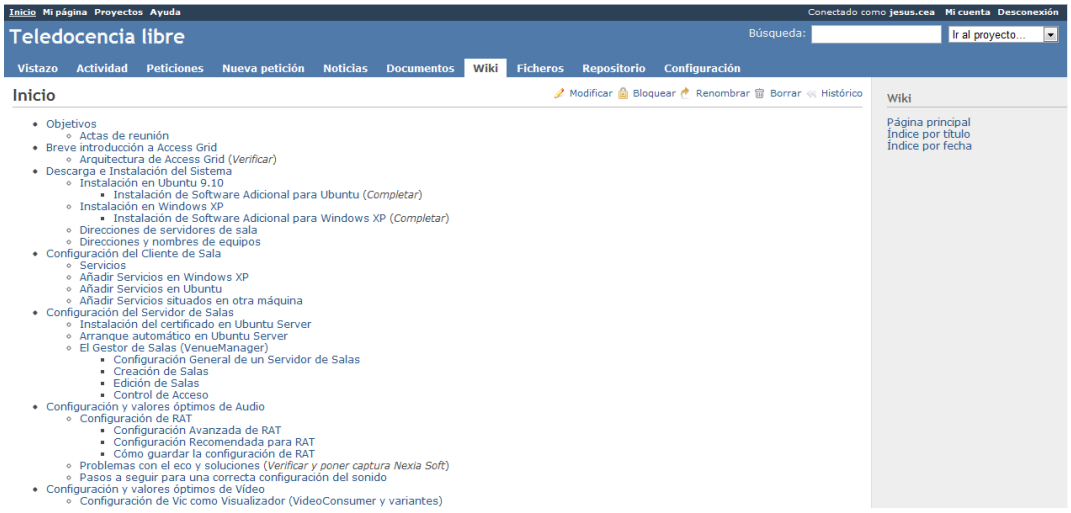


Figura C.5: Redmine

Toda información considerada como importante y toda la documentación generada sobre *Access Grid* se alojaba en la *Wiki* de *Redmine*. Al ser una aplicación web, tiene todas las ventajas de ésta, tal como que dicha información se puede editar desde cualquier lugar.

Además de la Wiki, Redmine tiene otras características y funciones:

- **Sistema de control de versiones (*Svn*):** De hecho, todo el código fuente de desarrollo para *Access Grid* se encuentra en dicho repositorio.
- **Subir archivos o imágenes**
- **Herramientas para Planificación y Gestión de Proyectos:** Tales como desarrollo de *Diagramas de Gannt*, gestión de tareas y objetivos, etc.
- **Gestión de noticias**: Para alojar cualquier noticia en relación al proyecto en desarrollo

En nuestro caso, la dirección web es la siguiente: `http://softwarelibre.uca.es:9000/redmine/wiki/teledocencia`. Aunque para acceder a la información es necesario disponer de una cuenta, hasta que se libere toda la información.

Apéndice D

Eventos Realizados

A continuación, en este capítulo, a modo de demostración del buen trabajo realizado y del correcto funcionamiento de las Salas de Teledocencia, se mostrarán una serie de eventos que se han llevado a cabo a lo largo de estos meses.

D.1. Máster de Agroalimentación

El Máster de Agroalimentación es un Máster impartido por el profesor Carmelo, en el Campus de Jerez. Fue el primer evento que se realizó en la nueva Sala de Teledocencia de Jerez utilizando, en lugar de *Access Grid*, el sistema *Adobe Connect*, a petición de Carmelo.

Fue el evento que más incidentes tuvimos, ya que se inauguró con la Sala de Teledocencia de Jerez recién instalada y ocurrió los problemas que se comentaron en la respectiva sección (problemas de sonido principalmente). Se tuvo que parar momentáneamente el máster para solucionar el problema.

Sin embargo, una vez reanudado el máster, se realizó sin ningún tipo de problema y el profesor Carmelo acabó muy satisfecho. Se pretende que en el próximo curso se intente impartir las clases utilizando, en lugar de *Adobe Connect*, el sistema *Access Grid*.

D.2. Reuniones con las Universidades Andaluzas

Desde, aproximadamente, el mes de Febrero, semanalmente, todos los Jueves se han llevado a cabo una serie de reuniones a través de *Access Grid* entre todas las Universidades Andaluzas.

Estas reuniones han servido para compartir el conocimiento entre todos nosotros, cada uno descubría algo nuevo y lo mostraba al resto, desde saber configurar el sistema *Access Grid*, a saber utilizar algún equipamiento de la Sala, o, en nuestro caso, mostrar el desarrollo de software realizado, que quedaron muy sorprendidos y nos felicitaron.

Al comienzo había un poco de descontrol, ya que cada uno emitía a un códec diferente y había muchos problemas con el sonido. Sin embargo, actualmente, se han establecido algunos protocolos a seguir, como utilizar el códec H.261, y, si hay problemas de sonido, silenciarnos todos y emitir uno a uno para saber quién causa el problema, y se realizan reuniones con total normalidad y estable.

La Universidad de Cádiz ha sido felicitada por la gran calidad de imagen y la gran calidad de sonido, ofreciendo siempre una gran estabilidad y sin provocar ningún tipo de problema.

Figura D.1: Reunión entre las Universidades Andaluzas

En la imagen anterior se puede apreciar el **gran número de ventanas de vídeo** mostrados en pantalla, ofreciendo una gran calidad de vídeo y sin producir ningún tipo de pixelación.

Actualmente se realizan actas de reuniones y se establecen nuevos objetivos, propuestas, nuevas ideas y futuros proyectos donde cada uno puede colaborar.

Por otro lado, los Vicerrectores de todas las Universidades Andaluzas también se han reunido varias veces a través de sesiones de *Access Grid*. Se trataba de reuniones donde, antiguamente, quedaban en algún lugar físico, con los gastos que ello conlleva, desde costes del viaje, costes de alojamiento, dietas, etc.

Ahora, se reúnen cada uno desde sus Salas de Teledocencia o despachos (no todas las Universidades Andaluzas disponen de Salas de Teledocencia) a través del sistema *Access Grid* ahorrándose dichos costes.

Todas las reuniones llevadas a cabo se han realizado con éxito. Antes de comenzar la reunión, nos reuníamos para preparar los sistemas y asegurarnos de que todo el mundo tenía bien configurado su sistema de *Access Grid* y, si había algún problema, se corregía lo antes posible.

Actualmente se realizan las reuniones sin utilizar las Salas de Teledocencia, sino en sus respectivos despachos. Cada uno dispone de un PC equipado de una webcam de gran calidad y un sistema de microfonía con cancelación de eco. Se ha instalado en esos PCs el sistema *Access Grid* y se ha configurado según el equipamiento instalado. Desde entonces, los Vicerrectores se reúnen desde sus despachos a través de *Access Grid* ofreciendo un resultado muy bueno.

D.3. Máster de Gestión Portuaria

En el Campus de Puerto Real se reanudó un Máster de Gestión Portuaria. Este Máster comenzó al poco tiempo de inaugurarse las Salas de Teledocencia, pero, debido a que las Salas no estaban optimizadas y no había personal cualificado para utilizarlas, hubo multitud de errores que obligaron a suspender, temporalmente la utilización de dichas Salas.

Ahora que se ha configurado y optimizado las Salas de Teledocencia, se optó por reanudar este Máster y comprobar que, efectivamente, todo funcionaba correctamente.

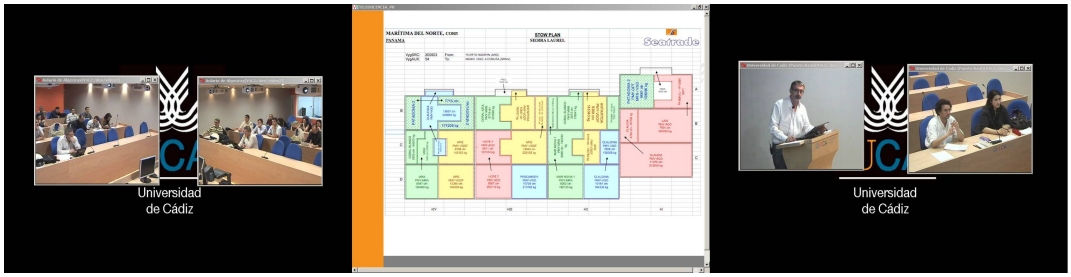


Figura D.2: Máster de Gestión Portuaria

En dicho Máster se utilizó el sistema *Access Grid* y se estableció una conexión con el Campus de Algeciras. Se impartieron tres clases, donde en cada una de ellas, algunas clases el profesor se encontraba en el Campus de Puerto Real y los alumnos se encontraban en los Campus de Puerto Real y Algeciras, y otras clases el profesor se encontraba en el Campus de Algeciras. Ésto fue interesante ya que se probó las dos partes del Máster, es decir, el lado del Profesor, donde había que preparar el equipo para transmitir las transparencias a Algeciras, además de preparar el enfoque de cámaras y poner las ventanas de vídeo oportunas en pantalla. Y el lado del alumno, donde había que conectarse al equipo de Algeciras remotamente para visualizar las transparencias, además de enfocar a los alumnos y distribuir las ventanas de vídeo oportunas.

Los resultados fueron muy satisfactorios y es logró finalizar el Máster, recibimos las felicitaciones de todo el equipo formado por dicho Máster.

D.4. Meeting sobre Física Aplicada

A modo de curiosidad, había un profesor de Física que ya utilizaba el sistema *Access Grid* desde su despacho. Utilizaba dicho sistema para asistir a varias sesiones y reuniones para hablar sobre Física Aplicada con multitud de personas que estaban en diversos países del mundo.

Sin embargo, debido a la instalación del nuevo Centro de Procesamiento de Datos, las redes cambiaron y, desde entonces, está teniendo problemas para poder utilizar *Access Grid* desde su despacho.

Dicho profesor no sabía la existencia de las Salas de Teledocencia, así que, mientras solucionaban el problema, cuando se enteró de que había una Sala de Teledocencia con sistema *Access Grid*, solicitó una reserva para utilizarla para una reunión que tenía urgente sobre Física Aplicada.

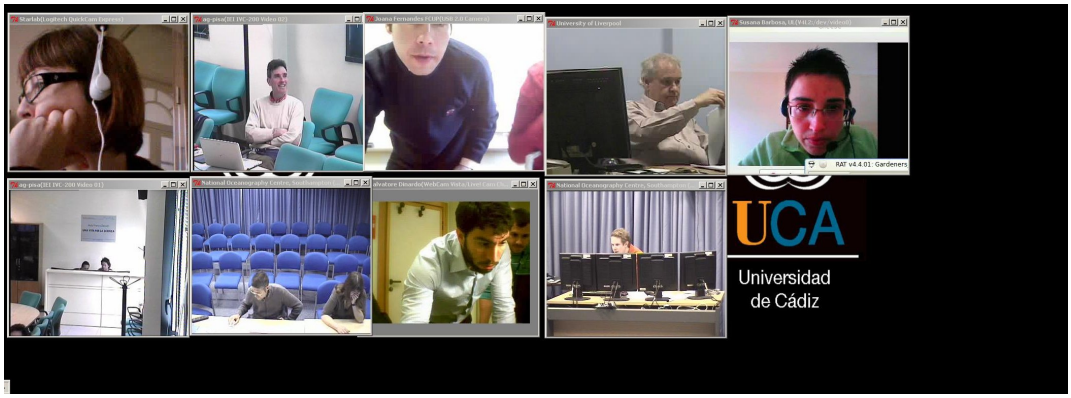


Figura D.3: Reunión de Física Aplicada

Esta reunión duró cuatro horas y fue todo un éxito. De hecho, fue curioso ver al resto de participantes, que se encontraban en lugares tan dispersos como en Manchester, Londres, Berlin, Chicago, etc, y, alguno de ellos no disponían de un equipamiento aceptable para utilizar *Access Grid*, provocando alguno de ellos eco, sonido distorsionado y una mala imagen.

El profesor nos felicitó por la gran calidad que tiene la Sala y por la labor realizada para que la reunión se llevase a cabo.

D.5. Workshop Image Analysis

El 29 de Abril se celebró un *Workshop* titulado *Image Analysis: Open Source Software*, cuya temática fue el análisis de imagen, orientado a la biología celular y microscopía electrónica, y empleando software de código abierto. El evento fue organizado conjuntamente por la Sociedad Española de Biología Celular [1] y la Universidad de Cádiz.

En el Workshop asistieron varias personas que realizaron diversas pruebas y estudios utilizando diversos programas, como *ImageJ*, *ImageSurfer* o *CellProfiler* y, cada uno, mostraban los resultados obtenidos. El Workshop fue retransmitido a la Universidad de Barcelona, quienes únicamente participaron para preguntar dudas acerca de las ideas expuestas, y, además, se retransmitió en vivo por streaming en tiempo real a través de Internet, pudiéndolo visualizar cualquier usuario. Este acto empezó a las 9.30 de la mañana y terminó a las 20.30 de la tarde.

La principal experiencia obtenida a partir de este Workshop fue demostrar que el sistema *Access Grid* fue estable. El sistema fue capaz de aguantar más de 10 horas sin tener ningún problema de estabilidad ni de ningún tipo.

Además, la Universidad de Barcelona desconocían completamente el sistema *Access Grid*, así que, a partir de varias indicaciones, consiguieron un PC con una webcam y nosotros, remotamente, a través del software *VNC* fuimos capaces de instalarles el sistema *Access Grid* y, además, configurarlo correctamente. Tuvimos un problema de sonido, ya que el sonido ”rebotaba” a nuestra Sala de Teledocencia (lo que hablábamos, al poco tiempo después lo oíamos en nuestra Sala). Accediendo al equipo de la Universidad de Barcelona, comprobamos que había una mala configuración de la tarjeta de sonido y lo solucionamos.

[1] `http://www.sebc.es/`

Bibliografía

[Agr10] *Página Oficial de Access Grid* [http://www.accessgrid.org/]

[Agd10] *Página Oficial de Access Grid para desarrolladores* [http://www.accessgrid.org/developer]

[Vis05] *Grupo de la Universidad de QueensLand (Australia) especializado en visualización avanzada, ciencias de la computación y colaboran en otros proyectos/tecnologías, entre ellas, Access Grid* [http://www.vislab.uq.edu.au/research/accessgrid/background.html]

[Ssl10] *Documentación de OpenSSL* [http://www.openssl.org/docs/apps/x509.html]

[Pms04] *Programmers Manual Shared Applications* [http://www.mcs.anl.gov/research/projects/accessgrid/documentation/SHARED_APPLICATIONS_MANUAL/ProgrammersManual_SharedApplicationsHTML.htm]

[Aws04] *Access Grid Workshop* [http://www.mcs.anl.gov/research/projects/accessgrid/documentation/tutorial/AGTk_2.4/index.htm]

[Pyt10] *Página oficial de Python* [http://www.python.org/]

[Ipy04] Mark Pilgrim - Francisco Callejo Giménez. *Inmersión en Python* [ISBN: 978-1590593561] [http://diveintopython.org/]

[Pop10] *Página oficial de la librería Poppler* [http://poppler.freedesktop.org/]

[Dop10] Valdoc *Documentación de la librería Poppler* [http://www.valadoc.org/poppler-glib/Poppler.Document.html]

[MaJ08] Mark Summerfield - Jasmin Blanchette *C++ GUI Programming with Qt 4 (2nd Edition)* - Prentice Hall [ISBN: 978-0132354165]

[Mar07] Mark Summerfield - *Rapid GUI Programming with Python and Qt* - Prentice Hall [ISBN: 978-0132354189] [http://books.google.es/books?isbn=9780132354189]

[Pqw10] F.-A. Bourbonnais. *PyQt4 on Windows* [http://v6sa.itcollege.ee/wiki/?page=PyQt4]

[Owe2006] Mike Owens - *The Definitive Guide to SQLite* - Apress [ISBN: 1-59059-673-0] [http://books.google.es/books?isbn=1590596730]

[Slt06] Peyton McCullough. *Using SQLite in Python* [http://www.devshed.com/c/a/Python/Using-SQLite-in-Python]

[Wxp10] *API de wxPython* [http://www.wxpython.org/docs/api/wx-module.html]

[Wik10] *Documentación de wxPython* [http://wiki.wxpython.org/]

[pqt10] Riverbank Computing Ltd - Nokia. *PyQt's Classes* [http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/classes.html]

[Nat06] Natalia Costas Lago. *Access Grid: Nuevos entornos de colaboración* [http://www.cesga.es/component/option,com_docman/task,doc_download/gid,291/Itemid,13/lang,gl/]

[Aga06] *Administration Guide to Access Grid* [http://www.med.cmu.ac.th/secret/mis/eiu/informatics/GridComp/TGCC2006/Acciss%20Grid/Administration%20Guide%20to%20Access%20Grid.pdf]

[Bel06] Jason Bell. *Successful Access Grid Use Cases* [http://www.aarnet.edu.au/library/Successful-Access-Grid-Use-Cases-Jason_Bell.pdf]

[Gro06] Michael Grobe. *The Alliance Access Grid* [http://www.greatplains.net/activities/meetings/meeting-20010419/presentations/MichaelGrobe/AG-for-UMKC.ppt]

[Sus06] Susanne Lefvert. *Developing Shared Applications* [http://www.mcs.anl.gov/research/projects/accessgrid/documentation/tutorial/AGTk_2.4/SharedApps/SharedApplications.ppt]